Computer Vision System for Automated Quality Inspection at Tractor Engine Assembly Line

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology
in
Communication System Engineering

by

Prasad Vinod Chavan (MT23CMN002)

under the guidance of

Prof. K. M. Bhurchandi
Professor, Department of ECE, VNIT Nagpur

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

June 2025

Computer Vision System for Automated Quality Inspection at Tractor Engine Assembly Line

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology
in
Communication System Engineering

by

Prasad Vinod Chavan (MT23CMN002)

under the guidance of

Prof. K. M. Bhurchandi
Professor, Department of ECE, VNIT Nagpur

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

June 2025

© Visvesvaraya National Institute of Technology (VNIT) 2025

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

I, **Prasad Vinod Chavan**, hereby declare that this dissertation titled **"*Computer Vision System for Automated Quality Inspection at Tractor Engine Assembly Line*"** is carried out by me in the **Department of Electronics and Communication Engineering** of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution / University.

Prasad Chavan

Date: _____

Certificate

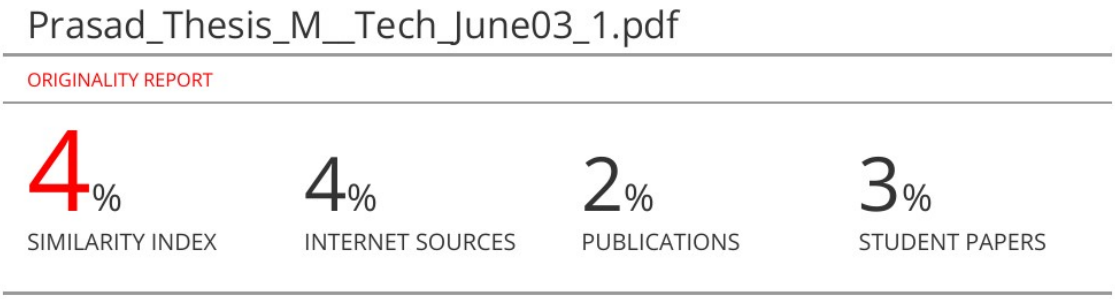
This is to certify that the dissertation titled **"*Computer Vision System for Automated Quality Inspection at Tractor Engine Assembly Line*"** is submitted by **Prasad Vinod Chavan (MT23CMN002)** in partial fulfillment of the requirements for the award of **Master of Technology in Communication System Engineering**, VNIT Nagpur. The work is comprehensive, complete, and fit for final evaluation.

Prof. Kishor Bhurchandi
Professor,
Electronics and Communication Engineering,
VNIT, Nagpur

Prof. Ashwin Kothari
Head of Department,
Electronics and Communication Engineering,
VNIT, Nagpur

Non-Plagiarism Certificate

I, **Prasad Chavan (MT23CMN002)** hereby declare that I am the sole author of the thesis entitled **Computer Vision System for Automated Quality Inspection at Tractor Engine Assembly Line** and that neither any part of the thesis nor the whole of the thesis has been submitted to any University or Institution for obtaining any degree/ diploma / academic award.It has been checked for plagiarism using **TURNITIN** software and the overlap is found within the prescribed limit. A summary of the report is given below:



Date: **Prasad Vinod Chavan**

Place: Nagpur **MT23CMN002**

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who supported and guided me throughout the course of this project.

First and foremost, I am deeply thankful to my project supervisor,
Prof. K. M. Bhurchandi, for his invaluable guidance, encouragement, and constructive feedback at every stage of this work. His expertise and insights played a crucial role in shaping the direction of this thesis.

I extend my heartfelt thanks to the **faculty**, **staff**, and **Prof. Ashwin Kothari**, H.o.D of the **Electronics and Communication Engineering, VNIT, Nagpur**, for providing the necessary academic environment and resources that made this project possible.

I would also like to extend my heartfelt gratitude to **Mr. Ajay Holey**, my Internship Manager, for his continuous guidance and encouragement throughout the project. I am deeply thankful to **Mr. Anil Bhakuni** and **Mr. Sandeep Jadhav**, whose mentorship as project guides played a crucial role in shaping this work. I also wish to acknowledge the entire Advanced Manufacturing Technologies team and John Deere R&D lab team, especially **Mr. Amit Patil** and **Mr. Prasad Patil**, for providing the necessary resources, support, and a highly collaborative environment. The hands-on exposure to real-world industrial systems and the opportunity to address practical engineering challenges greatly enhanced my technical and professional development.

A special thanks to my **friends**, **peers** and **colleagues** for their constant motivation and helpful discussions. Their input often helped me view problems from new perspectives.

Lastly, I am grateful to my **parents**, **sisters** and **all family members** for their unwavering support, patience, and belief in my abilities throughout this academic journey.

This project has been a valuable learning experience and a stepping stone towards my future endeavors in automation, computer vision, and advanced manufacturing systems.

Prasad Vinod Chavan

MT23CMN002

ABSTRACT

In Traditional manufacturing practices, manual inspection in engine assembly lines is prone to errors, fatigue, and inconsistency, limiting scalability and reliability in high-volume manufacturing. This dissertation presents an automated vision-based inspection system aimed at minimizing human involvement while ensuring accurate, real-time fault detection for Quality Inspection at Engine Assembly Line.

The system integrates high-resolution industrial vision sensors as Keyence, with a Siemens S7 PLC and a UR10e collaborative robot, all connected via the PROFINET protocol for synchronized operation. It is capable of detecting faults such as missing components, rust, scratches, misalignment, improper fastening, machining defects, and label verification using OCR.

To address cost constraints in smaller manufacturing setups, a low-cost prototype was developed using a Raspberry Pi camera module. Captured images were analyzed using both traditional image processing techniques and deep learning models. The system implemented two object detection models: SSD-MobileNetV2-FPNLite and YOLOv11, trained using a custom labelled dataset. The overall inspection accuracy exceeded 93% in test environments, whereas humans used to achieve around 80% accuracy.

This work contributes to smart manufacturing and Industry 4.0 initiatives by delivering a modular, scalable, and cost-effective vision inspection framework. It supports easy integration into existing PLC-based automation lines and enables a smooth transition from manual to automated quality control systems.

LIST OF FIGURES

3.1 Hardware Architecture-block diagram 11

3.2 S7 1200 PLC 12

3.3 PLC Comparison 12

3.4 UR 10e co-bot 13

3.5 Keyence VS-L160MX 14

3.6 Keyence camera FOV details 14

3.7 Keyence In-built Models 15

3.8 Raspberry-pi 4 model B board[1] 16

3.9 R-pi v2 camera[1] 17

3.10 PROFINET Architecture [2] 18

3.11 MQTT publish subscribe model[3] 19

3.12 2 Way communication between PLC, Robot and Camera 20

4.1 Keyence Camera Inbuilt Tools and Capabilities 24

4.2 Histogram Equalization 25

4.3 Adaptive Thresholding 26

5.1 Part presence/absence 29

5.2 Engine Surface Scratches 30

5.3 Machining Hole Offset Detection 30

5.4 Edge-cut part sample 31

5.5 Edge-cut part inspection 31

5.6 Dented part sample 32

5.7 Dented part inspection 32

5.8 Loose Part Installation 33

5.9 Measurement and Dimensioning 33
5.10 OCR 34
5.11 Connector wiring harness 34
5.12 Algorithmic Flow for Inspection 35
5.13 Area-Based detection 36
5.14 Shape-Based detection 37
5.15 Machining hole offset 38
5.16 Calculated pixel distance between centers 38
5.17 Engine surface damage/scratches 39
5.18 Rusty parts detection 39
5.19 Measurement and dimensioning approach 40
5.20 Measurement and dimensioning 40
5.21 OCR using EasyOCR 41
5.22 Detected Barcode using pyzbar 41
5.23 Barcode decoded data 41
5.24 Dataset images for model training 42
5.25 SSD-MobileNet object detection 45
5.26 Confusion Matrix(values in %) 45
5.27 YOLOv11 object detection 46
5.28 YOLOv11 Architecture [4] 47
5.29 mAP 48
5.30 Confusion Matrix(values in %) 48

LIST OF TABLES

5.1 Models performance comparison 43

6.1 Accuarcy of Keyence Vision System for various defect types 50
6.2 Inspection Accuracy Using Raspberry Pi and Computer Vision Algorithms 51
6.3 Performance Metrics of Deep Learning Models for object detection and classification 52
6.4 Comparative Analysis of Inspection Approaches 52
6.5 Defect-wise suitable inspection method 52

Contents

Abstract i

List of Figures ii

List of Tables iv

1 Introduction 1

1.1 Background . . . 1
1.2 Motivation . . . 2
1.3 Problem Statement . . . 2
1.4 Objectives . . . 3
1.5 Scope of project . . . 3

2 Literature Review 4

2.1 Introduction . . . 4
2.2 Traditional Inspection Techniques . . . 5
2.3 Machine Vision in Manufacturing . . . 5
2.4 Deep Learning and Visual Inspection . . . 6
2.5 Defect Detection in Engine Assembly . . . 7
2.6 Embedded Vision Systems . . . 8
2.7 Summary . . . 8

3 System Design and Methodology 9

3.1 Introduction . . . 9
3.2 Project Description . . . 9
3.2.1 Detailed goals and objectives . . . 9
3.2.2 Proposed solution hardware architecture . . . 11
3.3 Technologies Used . . . 12
3.3.1 PLC: Siemens S7-1200 . . . 12
3.3.2 Co-bot: UR10e . . . 13
3.3.3 Vision sensors: Keyence, Raspberry Pi . . . 14
3.3.4 Communication Protocol: PROFINET, MQTT . . . 18
3.4 Hardware Integration . . . 19

3.4.1 Integration process with PLC and UR Co-bot 19
3.4.2 Network and communication setup 20
3.5 Summary 21

4 Implementation and Inspection Logic 22
4.1 Introduction 22
4.2 Defects Detected 22
4.3 Tools and Techniques Used 24
4.3.1 VS creator Tools 24
4.3.2 Image processing and Computer vision tools and libraries 24
4.4 Summary 27

5 Inspection Results 28
5.1 Introduction 28
5.2 Keyence Camera 28
5.2.1 Defect wise inspection results 28
5.3 Image processing and CV 35
5.3.1 Used techniques, tools and libraries 35
5.3.2 Defect-wise inspection results 36
5.4 Deep Learning approach 42
5.4.1 Dataset Preparation 42
5.4.2 SSD-MobileNet-v2-FPNLite Implementation and Results 43
5.4.3 YOLOv11 Implementation and Results 46
5.5 Summary 49

6 Results and Evaluation 50
6.1 Introduction 50
6.2 Vision System Accuracy(Keyence) 50
6.3 Raspberry Pi based Image processing and CV 51
6.4 DL model Performance(TensorFlow vs YOLO11) 51
6.5 Comparative Analysis 52
6.6 Summary 52

7 Challenges and Solutions 54
7.1 Introdution 54
7.2 Hardware Communication and integration challenges 55
7.3 Lighting variation and image quality issues 55
7.4 Camera Positioning and Field of view 56
7.5 Tools and Parameter sensitivity in Keyence software 56

7.6 Limited compute power for high-resolution, real-time inference 56
7.7 Thermal management of embedded controllers during continuous operation 57
7.8 Summary 57

8 Conclusion and Future Scope **59**
8.1 Summary 59
8.2 Benefits over Manual Inspection 60
8.3 Future Scope 60

References **62**

Chapter 1

Introduction

1.1 Background

In modern manufacturing industries, ensuring product quality and process efficiency is crucial, especially in highly complex and high-throughput environments such as engine assembly lines[5]. Any defect or oversight in the final stages of production can cause significant downstream issues, customer dissatisfaction, and increased warranty costs. Traditional visual inspection methods are highly dependent on manual labour, which is not only time-consuming but also inconsistent and prone to human error[6, 7]. As production volumes and quality expectations increase, the need for automated, reliable, and intelligent inspection systems has increased significantly.

Computer vision and machine learning have become powerful tools in this domain. These technologies can analyse visual data to detect anomalies, classify parts, and verify assembly conditions in real time. In particular, their application to end-of-line inspection tasks, such as verifying part presence or absence, detecting rust or damage, detecting misalignments, and validating machining operations, offers significant improvements in accuracy, speed, and repeatability.

This project focuses on developing a vision-based inspection system for end-of-line quality checks of the engine assembly. The aim is to minimise human involvement and improve reliability using low-cost hardware platforms and custom-trained models.

1.2 Motivation

The motivation for this work comes from direct industry observation during the internship period, where manual inspection was found to be inconsistent, slow, and often unreliable. As observed by Frustaci et al.[8], manual inspection methods not only introduce delays due to process halts but are also prone to inaccuracies, especially when geometrical compliance checks are required in constrained assembly environments. Machine vision systems are rapidly transforming quality control in manufacturing by enabling real-time, repeatable, and accurate inspection of product features such as geometry, texture, and color[5]. Manufacturers aim to meet higher quality standards and reduce rework and recall rates, so reliance on manual inspection presents a bottleneck. In addition, standard industrial vision systems are often expensive and rigid in terms of customization. Manual inspection in SMEs often results in long cycle times and high error rates, as also observed by Shaloo et al.[9], who highlighted that traditional visual inspection leads to increased costs and inconsistency in quality control processes.

This project was motivated by the possibility of using cost-effective embedded vision systems as Raspberry Pi combined with custom-trained models to perform intelligent defect detection. It aims to bridge the gap between affordability and functionality, enabling even midsize industries to adopt advanced inspection systems. Furthermore, hands-on experience with real engine components and production lines provided insight into actual industrial challenges, which this work attempts to address through a practical, data-driven solution.

1.3 Problem Statement

Manual inspection methods at the end of the engine assembly are inefficient and susceptible to inconsistency, fatigue, and human error. Common defects as missing components, rusted/damaged part, misalignment, improper connections, and machining error often go unnoticed, leading to failure in quality assurance. There is a great need for a low-cost, scalable, and intelligent inspection system that can automate the detection of such issues.

1.4 Objectives

The primary objectives of this work are:

- Design and implement a vision-based inspection system for detecting various types of defects at the end of the engine assembly line.
- To develop machine learning and computer vision algorithms capable of recognizing part presence/absence, rust, damage, misalignment, and other faults.
- To evaluate the performance of the proposed system using real-world data captured on site.
- Explore the use of Raspberry Pi or similar embedded vision platforms for cost-effective deployment.
- Provide a scalable foundation for integrating similar systems in other parts of the manufacturing process, such as welding inspection.

1.5 Scope of project

This work specifically addresses the inspection of engine components assembled in the final stage of the production line. It covers:

- Image acquisition using embedded cameras.
- Data processing and annotation of defects.
- Training in models for various defects.
- Real-time inference and defect classification.

The project does not focus on full-line automation but sets the stage for integrating inspection results into larger automation workflows.

Chapter 2

Literature Review

2.1 Introduction

This chapter provides a comprehensive review of existing research and technologies in the domain of automated visual inspection for manufacturing systems. It begins by highlighting traditional inspection methods and their limitations, followed by a detailed overview of machine vision techniques and their application in defect detection. The chapter further explores the role of deep learning in visual inspection, discusses current practices in engine assembly inspection, and evaluates the feasibility of embedded vision systems. The aim is to establish a strong foundation for the proposed work and to identify research gaps that this project seeks to address.

Despite the rise of automation, human inspection still plays a key role in specific quality assurance tasks[10]. Vision inspection systems have gone under many significant transformation with the advancements in machine learning and computer vision technologies. Machine vision helps manufacturers improve product consistency and reduce waste through automated inspections[11].The article emphasizes how vision systems contribute to efficiency and traceability in Industry 4.0 applications[10, 11]. With increasing demand for automation and consistent quality control, smart inspection systems have become essential, especially in manufacturing domains such as automotive engine assembly.

This chapter reviews the related literature on vision-based inspection techniques, defect detection in engine components, the role of embedded vision systems, and related applications in inspection systems. Automated inspection systems enable continuous monitoring and improve defect detection in manufacturing lines[10, 12]. Praxie discusses how

automation not only speeds up inspections but also ensures standardization and repeatability[12].

2.2 Traditional Inspection Techniques

Manual visual inspection has been the dominant approach in manufacturing quality assurance. Operators are trained to detect defects such as:

- Part presence/absence
- Surface damages (scratches, dents, rust)
- Alignment or orientation issues
- Incomplete or loose assemblies
- Dimensioning Errors
- Machining Errors

However, these methods suffer from:

- High dependency on operator experience
- Inability to maintain performance during long working hours
- Inconsistent defect feedback
- Limited scalability and lack of standardisation

Some systems have augmented trained manual inspectors with checklists, sensors, and gauges, but these still lack adaptability and intelligent decision-making.

2.3 Machine Vision in Manufacturing

As noted by Javaid [6], machine vision significantly enhances productivity and accuracy in manufacturing environments by eliminating human error and enabling real-time automated defect detection, which is a cornerstone of Industry 4.0 practices. Machine vision has become a key enabler for automated inspection.

Machine vision typically includes:

- Image acquisition using cameras
- Preprocessing (e.g., filtering, contrast adjustment)
- Feature extraction and classification
- Output interpretation and decision logic

Machine vision systems have achieved high accuracy in detecting surface defects, presence of parts, and verifying assembly completeness. Khan et al.[7] demonstrated that even simple image characteristics such as pixel thresholds, grayscale segmentation, and dimension analysis can deliver effective results for real-time defect detection in automated inspection setups.

Limitations of the earlier systems included rule-based logic, which was inflexible and often failed in cases of unexpected variations in defect types or say variable lighting conditions. Wen et al.[13] introduced a modular system architecture that divides quality inspection into sub-tasks like object recognition, defect analysis, and data logging—an approach that closely aligns with the layered inspection pipeline used in this project.

2.4 Deep Learning and Visual Inspection

With the rise of deep learning, especially convolutional neural networks (CNNs), visual inspection systems have evolved from rigid rule-based systems to intelligent and adaptive frameworks.

Recent research has demonstrated successful use of CNNs for:

- Object detection(YOLO, SSD)
- Image classification(defective or non-defective)
- Detecting anomaly in manufacturing processes
- OCR for label and nameplate reading

Deep learning enables:

- Learning from data rather than manual rule definition
- Handling complex defect patterns
- Enhanced resilience to noise and light condition variations

However, deploying these systems often requires powerful GPUs and computing infrastructure, which can be a costly solution for small or medium-sized industries.

2.5 Defect Detection in Engine Assembly

Inspection in engine assembly involves multiple parameters check:

- Fastener presence
- Orientation of components (e.g., injectors, manifolds)
- Surface quality (e.g., painted regions, rust)
- Machining precision (e.g., holes, alignment)
- Barcode or nameplate verification

Few publicly available research works specifically focus on end-of-line inspection for engine components using deep learning. Most industrial applications are proprietary (e.g., Bosch) and use expensive commercial vision solutions (e.g., Keyence, Cognex, ifm, etc).

Research gaps identified:

- Lack of open datasets for automotive engine defects inspection
- Limited work on cost-effective embedded solutions
- Few works addressing part-level fault classification in a unified vision pipeline[8]

2.6 Embedded Vision Systems

Low-cost embedded platforms such as Raspberry Pi have enabled affordable deployment of vision-based systems in constrained environments.

Recent studies show:

- Raspberry Pi can handle lightweight object detection models (e.g. MobileNet-fpnlite models, YOLO-n)
- On-device inference reduces dependency on cloud/GPU servers

Challenges include:

- Limited compute power for high-resolution, real-time inference
- Thermal management during continuous operation
- Integration with industrial networks (e.g., PLCs, MES systems)

2.7 Summary

This chapter reviewed various inspection methodologies and the progression from manual techniques to advanced machine vision and deep learning approaches.

Key takeaways include:

- Traditional inspection is prone to human error and lacks scalability.
- Machine vision significantly improves detection accuracy and repeatability in industrial settings.
- Deep learning enables adaptive, data-driven inspection, but often requires high computing resources.
- There is limited literature on affordable, embedded vision systems tailored for engine assembly inspection.

The proposed work addresses these gaps by leveraging Raspberry Pi-based hardware and lightweight detection models for a modular, cost-effective solution.

Chapter 3

System Design and Methodology

3.1 Introduction

This chapter outlines the architecture and technical foundation of the proposed vision-based automated inspection system. It begins by describing the goals and problem statement driving the need for automation in quality inspection. The chapter then details the hardware components, including the Siemens S7-1200 PLC, UR10e collaborative robot, and vision systems such as Keyence and Raspberry Pi. Communication protocols like PROFINET and MQTT are introduced to explain how devices exchange data in real-time. Additionally, integration workflows between hardware units are illustrated, establishing the groundwork for reliable and synchronized industrial inspection processes.

3.2 Project Description

During my internship at John Deere, I had the opportunity to work on several exciting projects that focused on advanced manufacturing technologies, particularly in the areas of automation and computer vision. Each project was designed to address specific challenges in the manufacturing process and aimed at improving efficiency, accuracy, and overall quality.

3.2.1 Detailed goals and objectives

This project was the development of proof of concept for a vision-based automated quality inspection system.

The current assembly inspection process relies heavily on manual dependency, which

can lead to several errors, including missing parts, incorrect alignment or orientation, cross-fitment issues, and damaged or bent components. The article emphasizes how vision systems contribute to efficiency and traceability in Industry 4.0 applications[11, 12]. These errors can significantly impact product quality and overall process efficiency, resulting in increased costs and decreased customer satisfaction. To address these challenges, there is a pressing need for an automated vision inspection system. Such a system would enhance accuracy and reliability in the inspection process, ensuring that each component meets the required standards and ultimately improves the integrity of the final product[12]. By implementing this automated vision inspection process, we can achieve greater precision and streamline the assembly workflow, leading to improved outcomes in manufacturing.

For the above problem statement, we proposed a solution as- The implementation of high-resolution cameras combined with advanced image processing algorithms allows for real-time inspections during the assembly process. By integrating this vision system with collaborative robots (co-bots), manufacturers can achieve a highly efficient and accurate inspection workflow. This seamless integration into existing assembly lines not only enhances the overall inspection capabilities but also promotes safe collaboration between humans and robots[10]. The result is a more streamlined operation that maintains high-quality standards while ensuring a safe working environment for all personnel involved.

Eliminating dependency on manual inspection is crucial for enhancing efficiency in the manufacturing process. By doing so, we can significantly reduce labour costs associated with traditional inspection methods. Furthermore, the implementation of automated systems leads to improvements in process accuracy, ensuring that each component meets quality standards consistently. This solution not only enhances product quality but also optimizes inspection time, allowing for faster throughput in assembly lines. Ultimately, these changes contribute to a more efficient and cost-effective manufacturing operation

3.2.2 Proposed solution hardware architecture

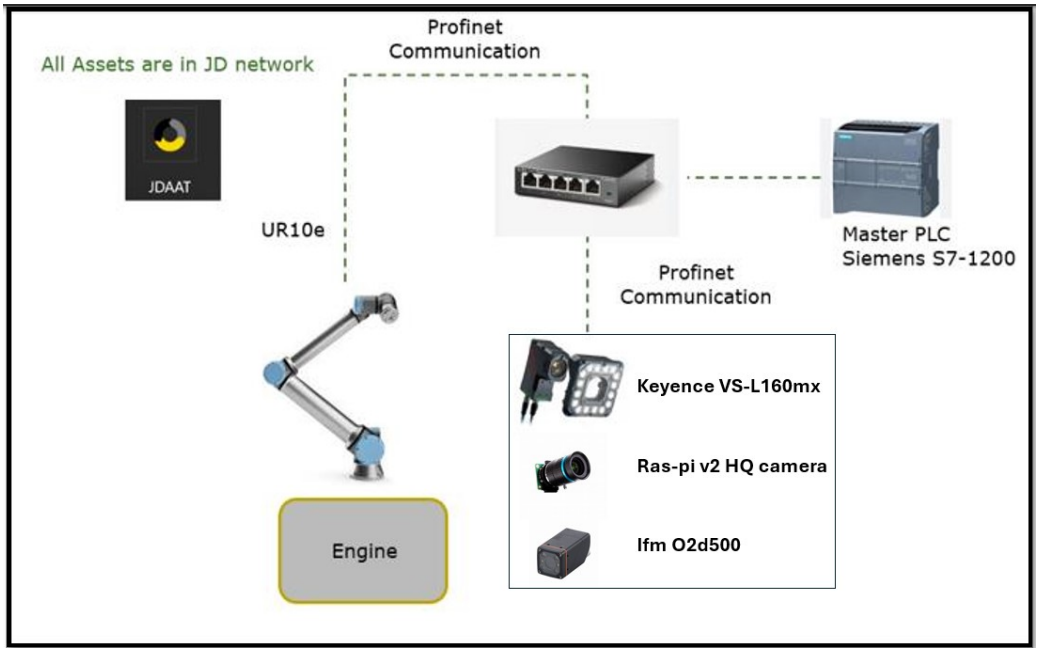


Figure 3.1: Hardware Architecture-block diagram

The above diagram illustrates the overall hardware architecture of the vision-based automated quality inspection system for engine assembly line. Key hardware components include the Siemens S7-1200 PLC, UR10e collaborative robot, and industrial vision sensors such as Keyence VS-L160MX and Raspberry-pi. All devices are interconnected via an Ethernet hub over the John Deere internal network infrastructure, enabling synchronized communication. The PLC serves as the central controller, facilitating real-time data exchange between the vision system and the robot through the PROFINET protocol. The vision sensors capture inspection images and communicate fault detection results back to the PLC, which in turn notifies the system about specific fault in that specific engine. This modular setup ensures reliable, deterministic, and scalable performance, suitable for continuous industrial deployment.

3.3 Technologies Used

3.3.1 PLC: Siemens S7-1200

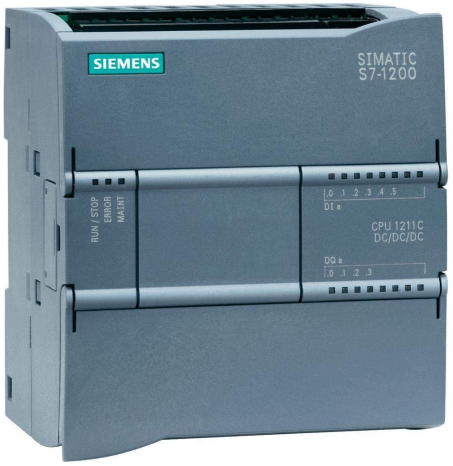


Figure 3.2: S7 1200 PLC

- Max transmission rate: 100Mbit /s.
- Input delay: 0.2ms to 12.8ms.
- Interface type: Profinet.
- Compact CPU with 8 DI and 6 DO pins with 24V DC.

PLC	Siemens S7 1200	Siemens S7 1500	Allen Bradley MicroLogix 1400	Mitsubishi FX5U
Advantages	-Cost effective for small to medium automation -Profinet communication -Built in web server for remote monitoring -Easy Integration with Siemens TIA portal	-High performance with faster processing. -Advanced diagnostics. -scalable for complex systems. -Integrated advanced safety features.	-User friendly software. -Built-in ethernet for communication. -Expandable I/O -Cost effective for small systems	-High speed processing. -Built-in Ethernet. -Good integration with Mitsubishi products -Scalable with modular design
Disadvantages	-Limited scalability for large systems -Lower processing powercompared to 1500 -Basic safety features	-More expensive -overkill for small to medium systems	-Not scalable for complex and large systems. -Expensive software licensing.	-More complex programming. -
Suitability for project	-Most Suitable for our project -Adequate for sensor integration, real-time monitoring and cost effective automation.	-Good for future upgrades. -Ideal for if project grows or include more complex robotic control. -Offers better safety.	-Good alternative for siemens 1200 -A simpler option with decent capabilities but may lack the advanced processing if required.	-Suitable for expansion. -High performance for mid-sized automation projects. -Good choice after s7-1500 for complex systems.

Figure 3.3: PLC Comparison

3.3.2 Co-bot: UR10e

It's a versatile, collaborative robotic arm developed by Universal Robots.Designed to work alongside human operators in wide range of industrial applications, including assembly, packaging, palletizing, and inspection. The UR co-bot is known for its flexibility, ease of use, and advanced safety features.

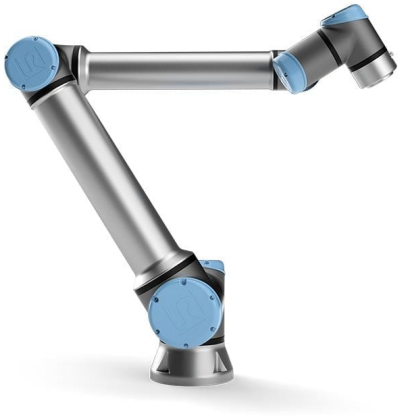


Figure 3.4: UR 10e co-bot

Key Features and Specifications:

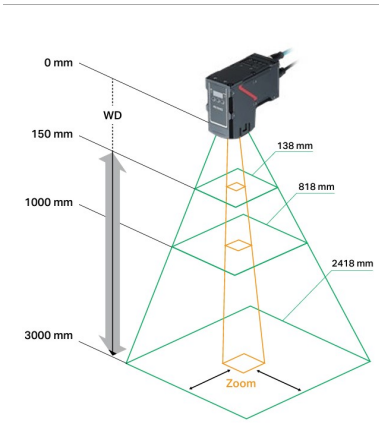
- **Payload Capacity**: The UR10e can handle payloads of up to 12.5 kg, making it suitable for various tasks.
- **Reach**: With a reach of 1300 mm, the UR10e can cover a large work area, allowing it to perform tasks that require extended reach and flexibility.
- **Repeatability**: The robot offers a repeatability of ±0.05 mm, ensuring precise and accurate movements.
- **Ease of programming**: It features an intuitive programming interface, allowing users to quickly set up and program the robot without extensive robotics expertise.

3.3.3 Vision sensors: Keyence, Raspberry Pi

Keyence VS-l160mx

Figure 3.5: Keyence VS-L160MX

- Keyence FOV details:



	Working distance	Field of view	Pixel resolution
VS-L160	150 mm	18 to 138 mm	12 to 95 µm
	1000 mm	94 to 818 mm	65 to 568 µm
	3000 mm	274 to 2418 mm	190 to 1679 µm
VS-L320	150 mm	25 to 138 mm	12 to 67 µm
	1000 mm	134 to 818 mm	65 to 399 µm
	3000 mm	390 to 2418 mm	190 to 1180 µm
VS-L500	150 mm	31 to 138 mm	12 to 56 µm
	1000 mm	162 to 818 mm	65 to 331 µm
	3000 mm	470 to 2418 mm	190 to 981 µm
VS-L1500	150 mm	55 to 138 mm	13 to 31 µm
	1000 mm	289 to 818 mm	65 to 185 µm
	3000 mm	839 to 2418 mm	190 to 549 µm

Figure 3.6: Keyence camera FOV details

- Capabilities and Models

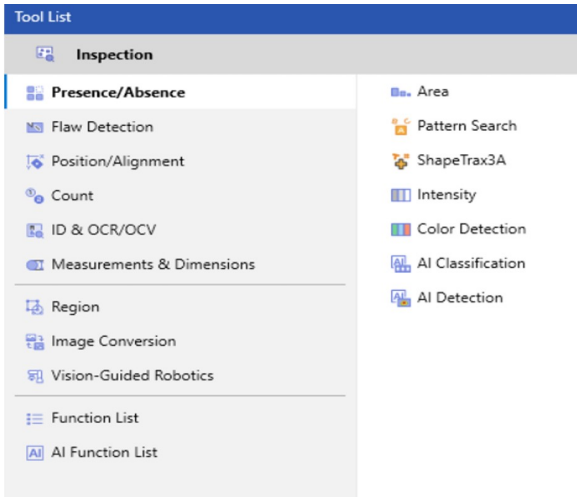


Figure 3.7: Keyence In-built Models

Keyence vision systems offer a comprehensive suite of tools tailored for a wide range of industrial inspection applications. These tools are categorized based on functional requirements such as presence/absence detection, flaw detection, alignment verification, optical character recognition (OCR), dimensional measurements, and AI-based classification. Each tool is designed to perform under varying lighting and environmental conditions, with user-configurable parameters for high accuracy.

For instance, the presence/absence tool verifies whether a component is correctly installed and present at right location, while the flaw detection tool identifies surface defects like scratches, dents, or cracks. The alignment tool ensures that parts are properly oriented, and the OCR module reads alphanumeric codes or labels for traceability. Dimensioning tools allow precise measurements of lengths, widths, and diameters, while the AI tool can be trained to recognize complex patterns and defects beyond rule-based logic. These modular tools enables the creation of flexible and application-specific inspection programs that can be easily adapted to different parts and processes in a manufacturing environment.

Raspberry-pi Kit

- R-pi 4 board model B The Raspberry Pi 4 Model B is a compact, low-cost, and

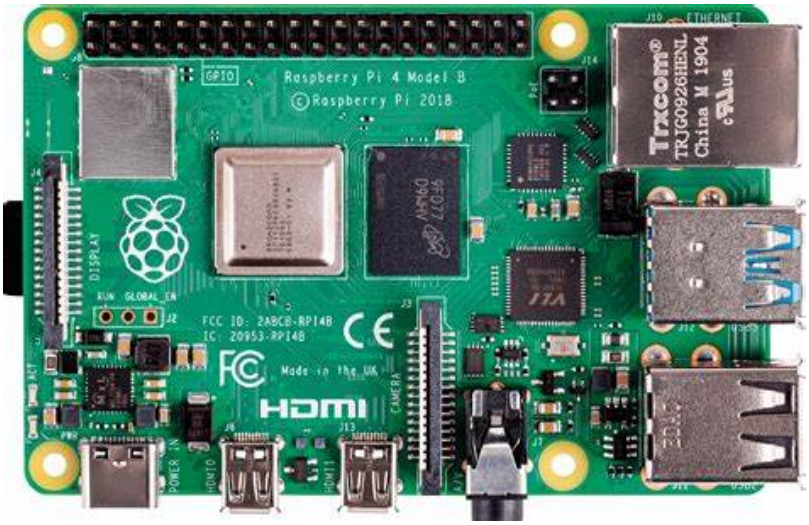


Figure 3.8: Raspberry-pi 4 model B board[1]

high-performance single-board computer widely used in embedded vision and IoT applications[1]. It features:

  - Processor: Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz
  - RAM Options: Available in 2 GB, 4 GB, and 8 GB LPDDR4 SDRAM variants (4 GB used in this project)
  - Connectivity: Dual-band 2.4/5.0 GHz Wi-Fi,

    Bluetooth 5.0

    Gigabit Ethernet

    2 × USB 3.0 ports and 2 × USB 2.0 ports
  - Display Interfaces: 2 × micro-HDMI ports (support up to 4K output)
  - Camera Interface: 1 × CSI port for camera module connection
  - GPIO: 40-pin header for interfacing with sensors, relays, and other I/O devices
  - Power Supply: USB-C power input (5V 3A recommended)

In this project, the Raspberry Pi was used as a compact vision-processing unit for capturing images, applying image processing algorithms using OpenCV, and running object detection models in real-time.

R-pi HQ v2 camera

The Raspberry Pi HQ Camera v2 is a high-resolution camera designed for use with the Raspberry Pi's CSI interface. It supports interchangeable lenses via a standard C/CS mount, making it ideal for industrial and research applications.

Figure 3.9: R-pi v2 camera[1]

- Sensor: Sony IMX477R stacked, back-illuminated sensor
- Resolution: 12.3 megapixels (4056 × 3040 pixels)
- Sensor Size: 1/2.3 inch
- Lens Mount: Compatible with C and CS-mount lenses
- Shutter Type: Rolling shutter
- Frame Rates: Up to 60 fps at lower resolutions (e.g., 1080p)
- Interface: 22-pin CSI ribbon cable to Raspberry Pi CSI port
- Additional Features: Manual focus and aperture control (via adjustable lens)

In the inspection system, this camera was used to capture high-quality, close-up images of engine parts under inspection [1]. Its flexibility in lens selection and resolution made it suitable for detecting small surface defects and performing accurate presence/absence checks.

3.3.4 Communication Protocol: PROFINET, MQTT

PROFINET

PROFINET (Process Field Net) is an industry-standard Ethernet-based communication protocol developed by Siemens for real-time data exchange between controllers (PLCs) and field devices like sensors and actuators. It enables deterministic and high-speed communication essential for time-sensitive industrial applications such as motion control, vision inspection, and robotics[2]. PROFINET supports both cyclic and acyclic data transmission, allowing consistent real-time control and device configuration simultaneously[2]. Its robustness and low latency make it ideal for synchronous operations in automation, such as signaling fault detection from a vision sensor to a PLC for initiating part rejection.

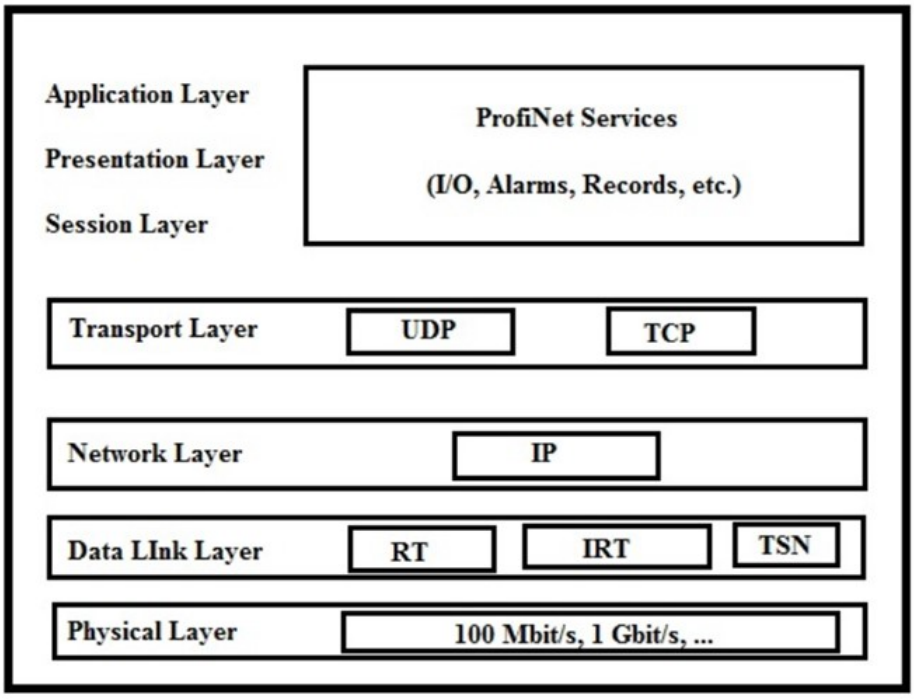


Figure 3.10: PROFINET Architecture [2]

MQTT

MQTT(Message Queue Telemetry Transport) is a lightweight, publish-subscribe messaging protocol designed for minimal network bandwidth and device resource usage[3]. It is especially useful in IoT applications where small devices like Raspberry Pi are used. MQTT decouples message producers (publishers) and consumers (subscribers), allowing asynchronous and scalable communication[3].

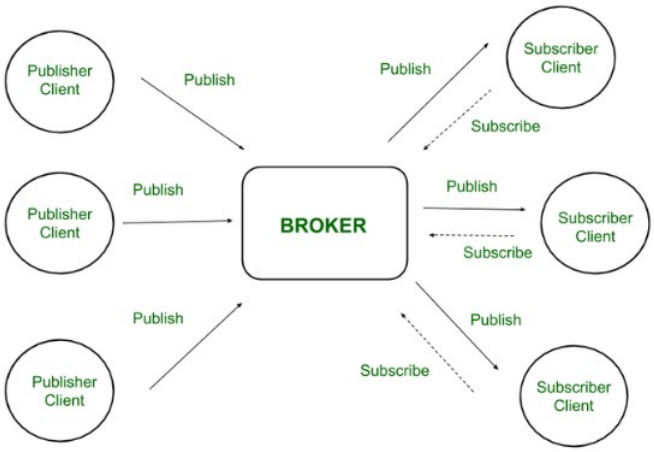


Figure 3.11: MQTT publish subscribe model[3]

3.4 Hardware Integration

3.4.1 Integration process with PLC and UR Co-bot

In this setup, two-way communication between the Siemens S7-1200 PLC, UR10e collaborative robot, and Keyence vision sensor was established via the PROFINET protocol using GSD (General Station Description) files. The GSD files for both the UR robot and the Keyence camera were imported into TIA Portal, enabling seamless hardware configuration and data exchange. Once integrated, each device was assigned a unique IP address, and appropriate I/O tags were mapped to PLC data blocks for real-time status monitoring and control.

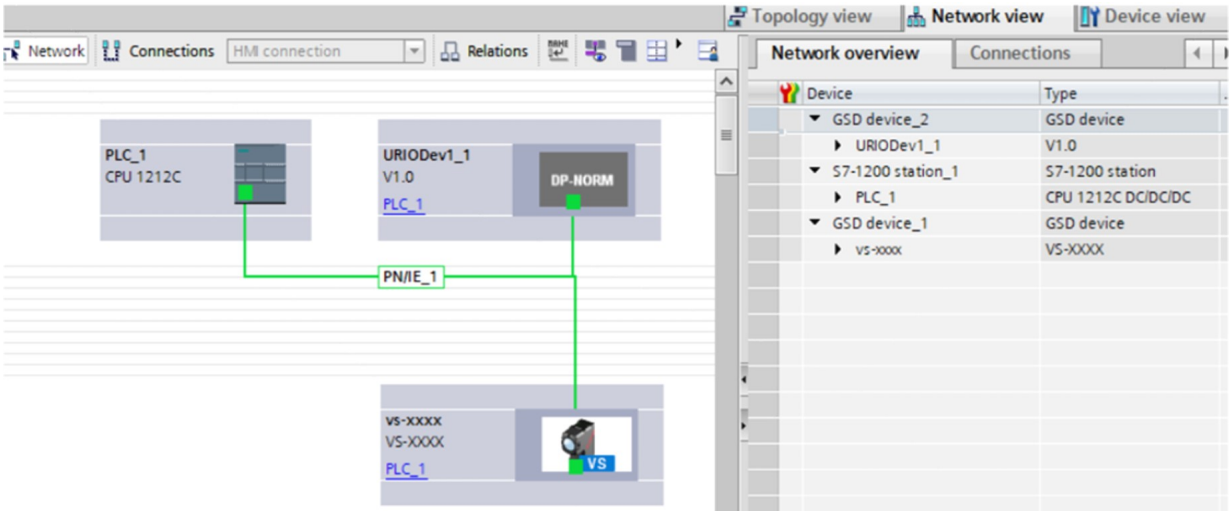


Figure 3.12: 2 Way communication between PLC, Robot and Camera

The robot sends trigger signals and receives inspection results, while the PLC acts as the central controller orchestrating commands between all devices. The successful GSD integration ensured stable, synchronized communication, essential for fault detection and inspection workflows. This configuration allowed high-speed, deterministic data flow between vision hardware and the control logic, forming a reliable automation framework.

3.4.2 Network and communication setup

Devices connected over industrial PROFINET for deterministic communication. Raspberry Pi interfaced with PLC via OPC UA and MQTT for scalable data flow.

PROFINET

In above project, PROFINET protocol is used to successfully communicate between the Siemens S7-1200 PLC, the vision systems (Keyence), and the UR10e Co-bot. This ensures that inspection results can be promptly processed and acted upon without delay. The structured nature of PROFINET also simplifies diagnostics and system monitoring, which is critical in industrial environments.

MQTT

In this system, MQTT is employed to facilitate data exchange between the embedded vision units (like Raspberry Pi) and higher-level applications or PLCs. It allows streaming of inspection results, logs, or alerts to remote systems for analysis or monitoring without heavy network overhead. This flexibility enables integration with cloud dashboards or remote control units for enhanced traceability and scalability. By using both PROFINET and MQTT, the system leverages the advantages of real-time control and scalable, lightweight data distribution.

3.5 Summary

This chapter presented the design and integration of a robust, automated inspection system using industrial-grade and embedded technologies. Key outcomes include:

- Defined system goals addressing the limitations of manual inspection processes.
- Described hardware architecture involving PLCs, robots, and vision sensors.
- Documented key technologies used: Keyence, Raspberry Pi, UR10e, Siemens PLC.
- Explained the role of PROFINET for real-time control and MQTT for scalable communication.
- Demonstrated successful integration and synchronization of devices through GSD-based configuration in TIA Portal.

The methodology discussed forms the technical backbone of the inspection framework, enabling real-time fault detection, process automation, and readiness for industrial deployment.

Chapter 4

Implementation and Inspection Logic

4.1 Introduction

This chapter focuses on the practical implementation of the visual inspection system and the various defect categories targeted during engine assembly inspection. It introduces the fault types considered, such as part absence, surface damage, machining deviations, and misaligned or loosely installed components. The chapter also outlines the tools and techniques applied, including both traditional image processing methods and Keyence's built-in vision tools. Each technique is discussed with reference to its suitability for industrial environments and effectiveness under variable lighting and resolution conditions. This chapter bridges the design methodology with the actual execution of fault detection strategies.

4.2 Defects Detected

- **Presence/Absence**

  Here we are interested in the detection of presence or absence of the various objects assembled on the engine as oil-cap, lubricant pipe, thermostat, cylinder, nameplate, barcode, etc.

- **Rusty parts**

  The assemled parts if rusty, then may lead to reduced life, degradation of performance, affect other good parts, etc. So the identification of such parts need to be done before they go on the actual machine assembly.

- **Damage(Scratch/ Dent)**

  The assembly of damaged parts on the machine may cause abnormal functioning of the other components as well. The assembly of damaged parts may further lead to fatal accidents like engine breakdown, etc.

- **Machining Hole offset**

  The offset in the machined parts, cutting of parts, or any profiling required on the machine, may cause misalignment or can be unfit for assembly of other parts near it, which can result in revert back of such engines.

- **Measurement and Dimensioning**

  The variation in dimensions of parts from the ideal ones, may cause misalignment or can be unfit for assembly of other parts near it, which can result in revert back of such engines.

- **Loose installation**

  The losse installation of any parts such as covers, flaps, oil caps, etc can cause to leakage or fatal spill of oil, or acidic liquids present in the engine.

- **OCR (nameplate/barcode)**

  The engine serial number and model number on the nameplate assembled on the engine must match with the actual serial and model number of the engine according to the JDAAT(John Deere Assembly Assist Tool) data. If it is not, then this can lead to wrong engines being sent to wrong lines of actual machine assembly.

- **Connector Wiring Harness**

  Inspecting the wiring harness and connectors is a critical step before the engine moves forward to final assembly on the machine or tractor. Any missing or incorrectly connected harness can disrupt circuit continuity, potentially leading to malfunction or failure of essential machine operations.

4.3 Tools and Techniques Used

4.3.1 VS creator Tools

Tool overview: Pattern Matching, Blob, Shapetrax, Profile Defect, Circle Detection, OCR are some of Inbuilt keyence tools commonly used by me in this inspection system development.

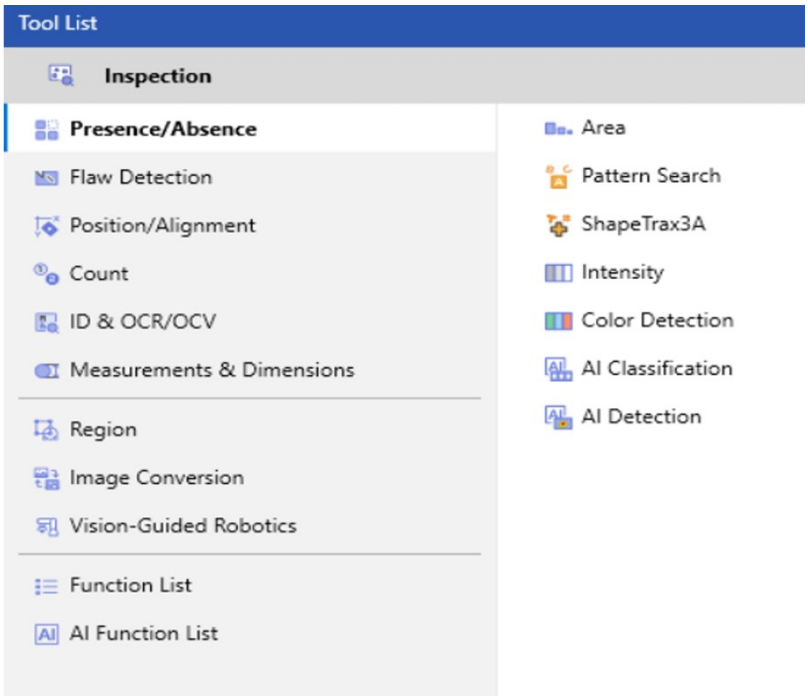


Figure 4.1: Keyence Camera Inbuilt Tools and Capabilities

As noted in[5], contour selection, region-of-interest filtering, and blob analysis are among the most widely adopted methods for real-time geometric verification in assembly-line inspection systems.

4.3.2 Image processing and Computer vision tools and libraries

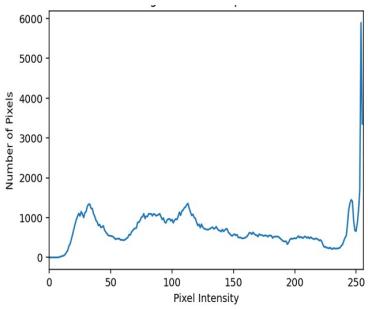
Sample image and flow for each defect Histogram Equalisation, Adaptive thresholding, Edge Detection, Contour detection.

Histogram Equalization

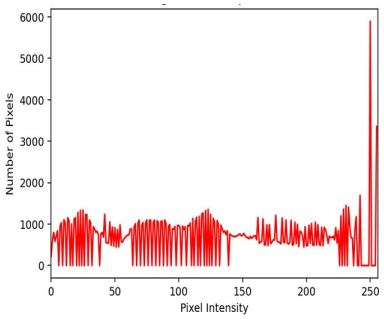
Histogram equalization is a contrast enhancement technique used to improve the visual quality of images by redistributing pixel intensity values. It transforms the histogram of an image to achieve a more uniform distribution, spreading out the most frequent intensity values. This helps in enhancing image features and improving visibility in poorly illuminated or low-contrast areas. Histogram equalization is particularly effective in highlighting edges and textures, which is crucial for computer vision tasks like fault detection. As you can observe in beside figure 4.2, The resulting image appears clearer and more balanced, making it suitable for consistent image processing across variable lighting conditions.

((a)) Original Image

((b)) After Histogram Equalization



((c)) Histogram of original image

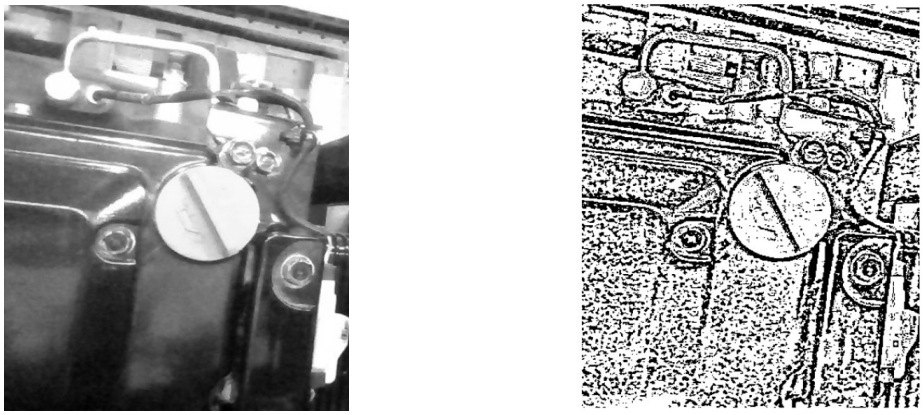


((d)) Histogram of Equalized image

Figure 4.2: Histogram Equalization

Adaptive Thresholding

Adaptive thresholding is a technique used to convert grayscale images to binary by computing a different threshold value for each pixel based on its local neighborhood. Unlike global thresholding, which uses a single value, adaptive thresholding is ideal for images with non-uniform lighting or shadows. It dynamically adjusts the threshold across the image, making it highly effective in handling varying illumination. This results in more accurate binary segmentation, allowing better distinction between foreground and background. Adaptive thresholding improves object detection, especially in complex or uneven lighting conditions commonly encountered in industrial inspection environments.



((a)) Original Image ((b)) After adaptive thresholding

Figure 4.3: Adaptive Thresholding

Edge Detection

Edges are regions in an image where there is a sharp intensity change, often corresponding to boundaries of objects. Canny Edge Detection is a widely used technique for identifying such edges with high accuracy and robustness. It involves unique double thresholding technique, which helps distinguish strong edges from weak ones. Canny's method minimizes false detections and ensures continuity of detected edges. It performs reliably even in noisy environments, making it ideal for tasks like contour-based defect detection and part shape verification in industrial inspection systems.

Khan et al.[7] demonstrated the effectiveness of using basic image characteristics, such as pixel count, shape, and grayscale thresholds—for object differentiation and defect identification, which closely aligns with the contour and thresholding methods implemented in our Raspberry Pi-based system. Recent advances have shown that convolutional neural networks (CNNs) and deep learning approaches like YOLO can outperform traditional methods in complex object classification tasks [5].

4.4 Summary

This chapter provided insight into the implementation aspects of defect detection using both Keyence tools and image processing algorithms on embedded platforms. Key outcomes include:

- Categorized and described multiple fault types relevant to engine assembly inspection.
- Demonstrated the use of Keyence VS Creator tools such as Pattern Matching, Blob, and Shapetrax.
- Explained fundamental vision techniques like histogram equalization, adaptive thresholding, and edge detection.
- Established the practical applicability of combining industrial sensors with computer vision libraries on Raspberry Pi.

These implementations form the core logic of the inspection system, enabling accurate real-time fault detection under industrial operating conditions.

Chapter 5

Inspection Results

5.1 Introduction

This chapter presents a progressive evaluation of three different inspection strategies for detecting defects in engine assemblies. It begins with results obtained from the industrial-grade Keyence vision system, utilizing its built-in tools such as Pattern Matching, Blob Analysis, OCR, and Profile Defect detection. The system offered high accuracy and speed, but at a significant cost. To develop a more affordable solution, traditional image processing and computer vision techniques were implemented using a Raspberry Pi v2 HQ camera, covering the same set of faults. While this setup provided cost efficiency, it showed limited reliability in object classification and presence/absence detection. To address these limitations, deep learning-based approaches were explored using SSD-MobileNetV2-FPNLite and YOLOv11, trained on custom datasets captured using the Raspberry Pi kit. These models improved accuracy, especially under variable lighting and part orientations, offering a scalable alternative to industrial sensors.

5.2 Keyence Camera

5.2.1 Defect wise inspection results

- **Part presence/absence**

In this Presence/Absence type of fault detection, we tried to detect various objects or parts on engine, as Oil cap, Lubricant pipe, Thermostat, Nameplate, Barcode, etc.

((a)) Cylinder Detection

((b)) Lubricant Pipe Detection

Figure 5.1: Part presence/absence

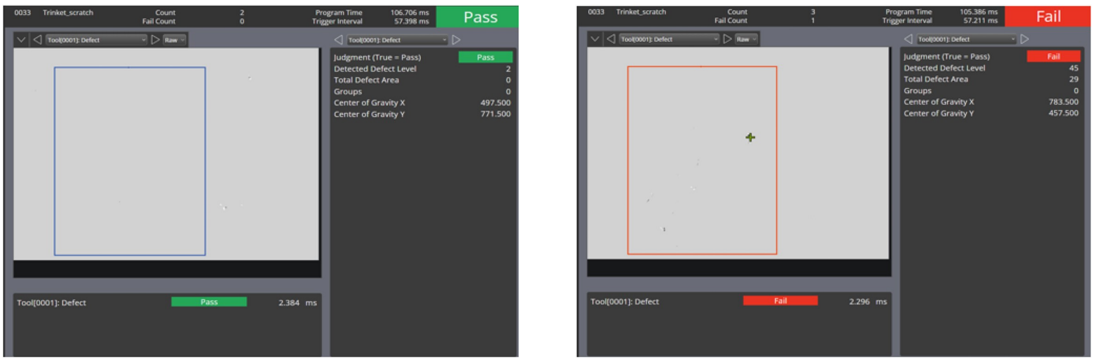
Here, above pasted are some of the results for object presence/absence trials. Basic approach we follow to design a program in VS creator:

1. Capture the image from a fixed distance ensuring the full part is within the camera's FoV. Apply autofocus and auto-brightness using Zoomtrax settings.
2. Reuse Zoomtrax settings for multiple parts if lighting and distance remain consistent. In VSL160mx, up to 2 image captures can be configured in one program.
3. Save the configured Zoomtrax and capture tools, then register the images for inspection.
4. For each inspection tool, select the corresponding image and proceed with fault detection tasks.
5. For object presence/absence, tools like Pattern Matching and ShapeTrax are used, allowing shape-based detection based on the geometry of the target part.

- **Part Damage(Scratch)**

Tools used: Flaw Detection, Defect, Profile Defect, Area, Blob.

The Damaged parts(surface scratch, dents, edge cut, etc) were identified using the above-mentioned tools.



((a)) Good condition ((b)) Not Good condition

Figure 5.2: Engine Surface Scratches

The Scrathes were identified by a defect tool, which takes the plane surface as the ideal condition, and if any irregularity present on the surface, is considered or highlighted as a defect. As the intensity of the defect will vary as compared to the intensity of the regular surface. So we adjusted the intensity in range of average of the black and white intensity. So the regular surface will be in gray shade and any irregularities present will be highlighted in white or black, as per our settings.

- **Machining Hole Offset**

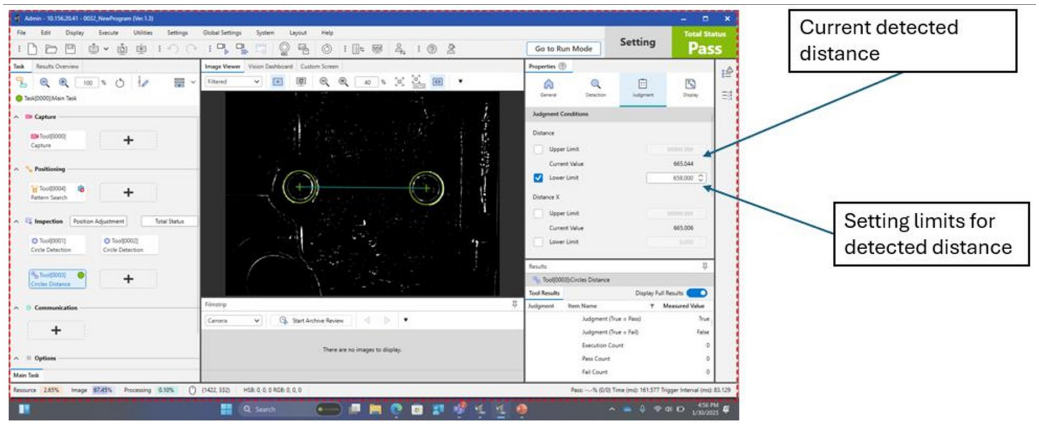


Figure 5.3: Machining Hole Offset Detection

Tools used: Circe Detection, Circle Distance.

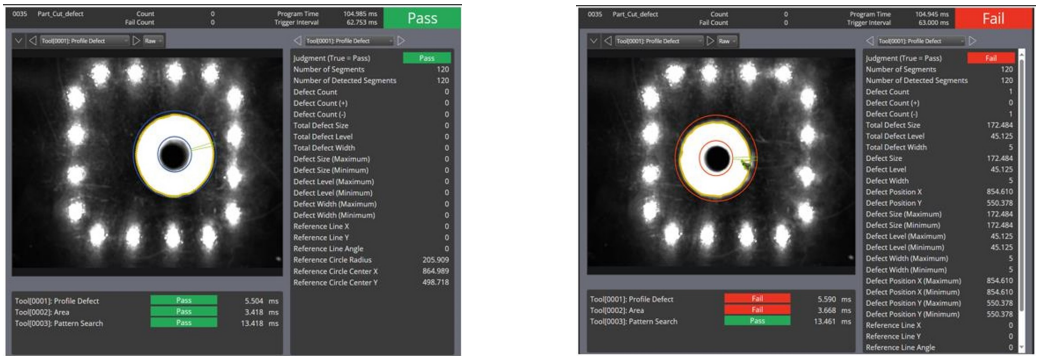
In inspecting this fault, we used two tools to execute it. Firstly, we did the Circle inspection, separately for both the circles using “Circle Detection” tool. As the 2 circles were detected, after that we used the “Circle distance” tool to detect the distance between

the 2 circles, we defined using Circle Detection tool. We can set threshold for the distance that can be considered as pass or fail according to our requirements of program.

- **Part Damage(Edge Cut)**

((a)) Good condition ((b)) Not Good condition

Figure 5.4: Edge-cut part sample



((a)) Good condition ((b)) Not good condition

Figure 5.5: Edge-cut part inspection

Tools used: Profile Defect

In this Fault detection, I used the Profile defect tool, to inspect if any edge cut is present in the part inspected. The Profile defect tool gives us in built provision to inspect some of the geometric shapes, like circle, rectangle, ring, etc. They can be used according to the requirement and shape of part to be inspected. Here In this defect we used the ring shape to detect the correctness of the part and inspect if there is any problem with the parts profile. If there is any irregularities in parts profile, then obviously it will not match the exact profile we expect.

- **Part Damage(Dent)**

((a)) Good condition ((b)) Not Good condition

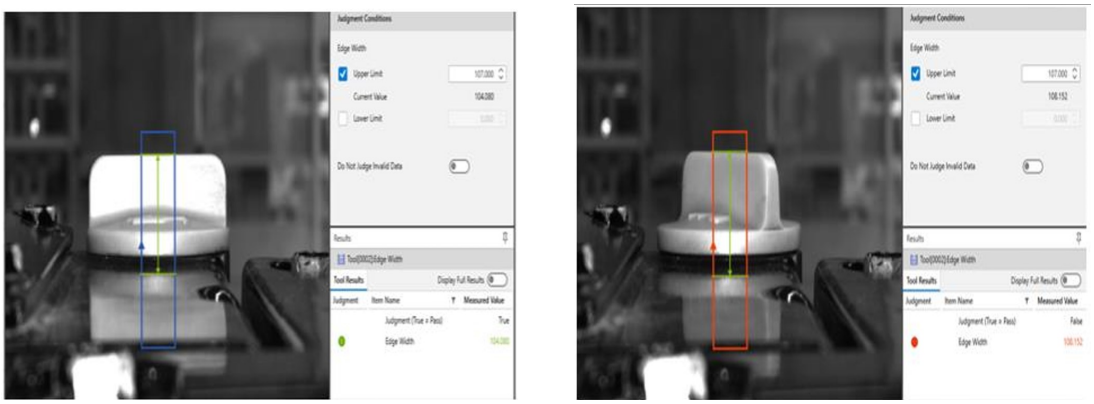
Figure 5.6: Dented part sample

Figure 5.7: Dented part inspection

Tools used: Area, Blob.

The Profile defect tool gives us in built provision to inspect some of the geometric shapes, like circle, rectangle, ring, etc. They can be used according to the requirement and shape of part to be inspected. Here In this defect we used the ring shape to detect the correctness of the part and inspect if there is any problem with the parts profile. If there is any irregularities in parts profile, then obviously it will not match the exact profile we expect.

- **Loose Part Installation**



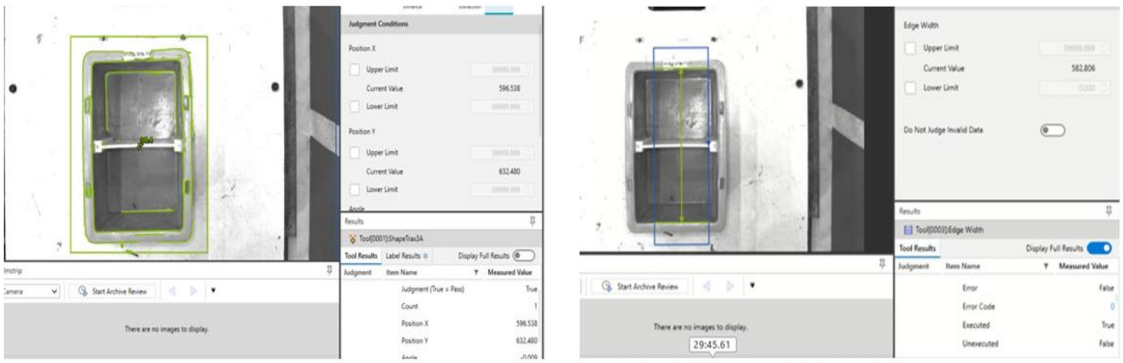
((a)) Good condition ((b)) Not Good condition

Figure 5.8: Loose Part Installation

Tools used: Edge width

In this inspection, we used the Edge width tool to detect the edge length. This edge length is nothing but the actual distance of the caps top edge from the specific edge. These edges are detected by the tool itself using various settings, like edge direction and the edge colour transition, like from dark to bright, bright to dark, or both.

- **Measurement and Dimensioning**



((a)) Rectangle detection ((b)) Edge width measurement

Figure 5.9: Measurement and Dimensioning

Tools used: Pattern matching, Edge width.

In this measurement inspection, we used the similar tool that we used in the previous inspection of the installation of loose cap and followed the same process.

- **OCR of Name-plate**

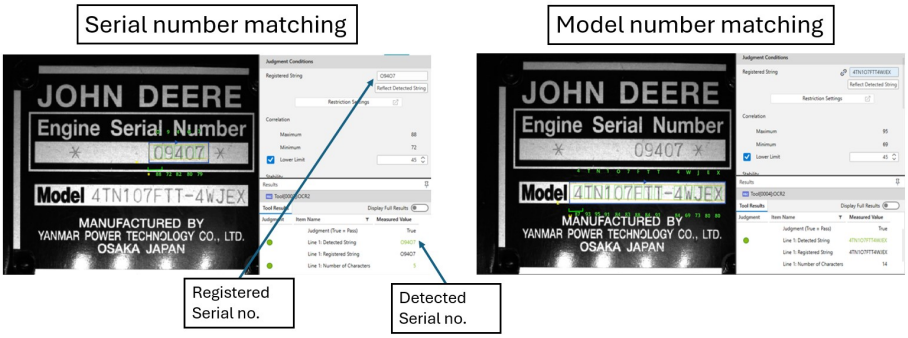


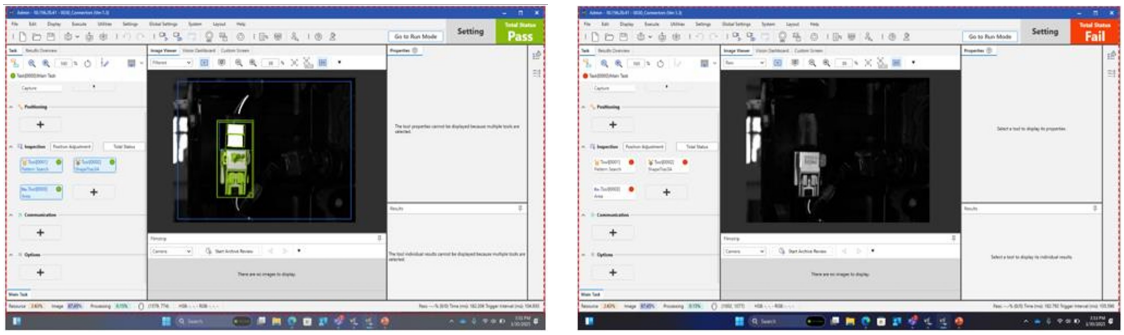
Figure 5.10: OCR

Tools used: OCR.

For the Nameplate, Barcode reading, there is an in-built tool of OCR in VS creator. This OCR tool can be used for regular OCR, 1D barcode reading, 2D barcode reading. For reading nameplate, we used regular OCR tool. We can also hide some of the characters from getting read, using masking tool. Also to avoid wrong character detection, we can put filters on the characters detected, like just read numbers, just read alphabets, or don't read any specific alphabet or number, etc. For Barcode, QR code reading we used 1D OCR and 2D OCRs.

- **Connector Wiring Harness**

The Connector presence/absence is detected using the Area tool.



((a)) Good condition ((b)) Not good condition

Figure 5.11: Connector wiring harness

5.3 Image processing and CV

Frustaci et al.[8] employed robust edge and morphological filtering techniques like Canny edge detection and median filtering, which are also applied in this project to isolate key fault regions such as scratches and hole offsets under varying lighting conditions. Rather than relying on computationally intensive filters like correlation or edge enhancement, Khan et al. [7] proposed a simplified and fast grayscale and RGB pixel-level analysis for real-time applications—an approach mirrored in our OpenCV-based solution for low-cost defect detection.

5.3.1 Used techniques, tools and libraries

Histogram Equalisation, Adaptive Thresholding, Edge Detection, Contour Detection.

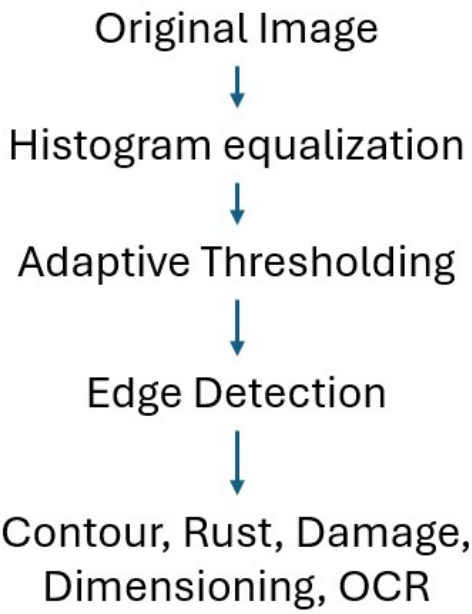


Figure 5.12: Algorithmic Flow for Inspection

The work by Wen et al.[13] emphasized the importance of image pre-processing steps such as noise removal and contrast enhancement in improving the robustness of edge-based defect detection under variable lighting conditions, which was also observed during our OpenCV-based fault detection trials.

5.3.2 Defect-wise inspection results

Here, I followed two approaches for the final contour detection, as the "**Area-based**" and "**Shape-based**" contour detection. We analysed both approaches, and from both, the area based approach performed better, but not up to the mark. Below results for both the inspection approaches.

- **Presence/absence**

Figure 5.13: Area-Based detection

Total Images: 114

Correctly predicted: 80

Accuracy: 70.2%

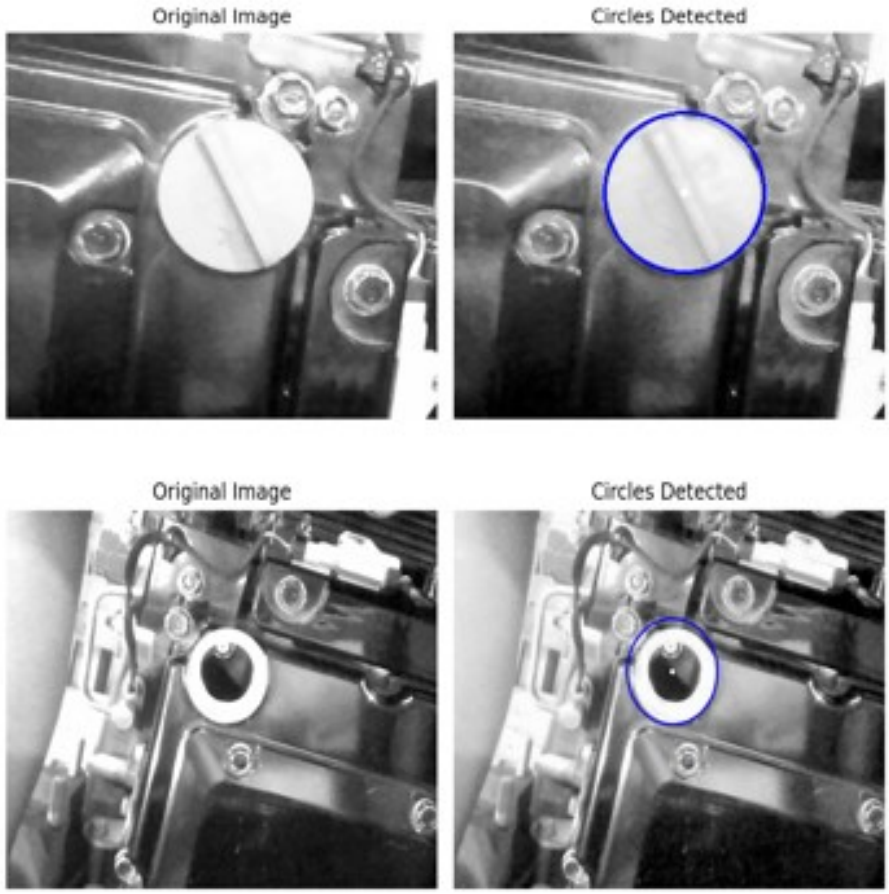


Figure 5.14: Shape-Based detection

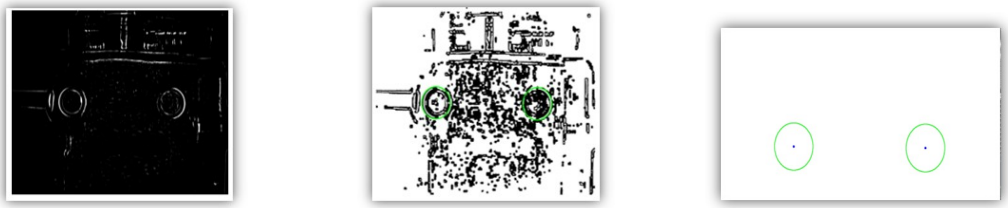
Total Images: 114

Correctly predicted: 69

Accuracy: 61%

- **Machining Hole Offset**

Similar to the approach we followed using Keyence inbuilt tools, I performed it step by step as we first detected the circles using the circular contour detection, then after both the favourable circles were detected, we calculated the distance between the center of both contour i.e. circles. Then the pixel distance of image width value(in pixels) is scaled to physical width of area covered in image, to calculate the actual distance between the circles.



((a)) Edge detection ((b)) Contour detection ((c)) Highlight center pts.

Figure 5.15: Machining hole offset

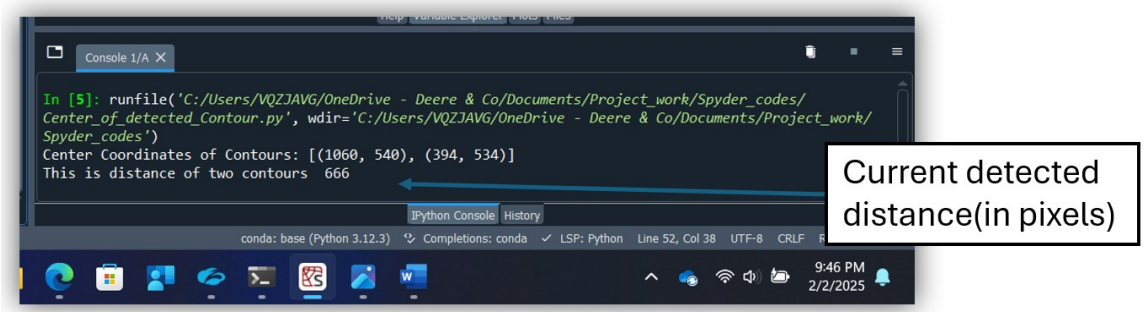


Figure 5.16: Calculated pixel distance between centers

Actual Distance: 140mm

Detected Distance: 138.75mm

Accuracy(in percentage): 99.1

Accepted Accuracy: 98.5(+2mm or -2mm)

- **Damaged Parts**

The Damaged parts(surface scratch, dents, edge cut, etc) were identified using the same algorithmic flow as Histogram equalization, Adaptive thresholding, Canny edge detection and the contour detection. As the intensity of the defect will vary as compared to the intensity of the regular surface. So the regular surface will be in background part colour and any irregularities present will be highlighted with green in original image, as per our settings.

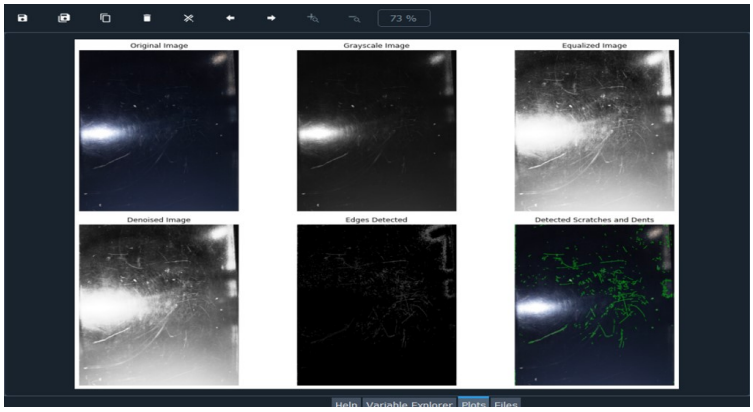


Figure 5.17: Engine surface damage/scratches

The Scrathes were identified as shown in above fig. 5.15, it takes the plane surface as the reference, and if any irregularity present on the surface, is considered and highlighted as a defect.

- **Rusted Parts**

Here, the Rusted parts were identified using the window frame for the colour intensity near the intensity of the actual rusted part i.e RGB range around [20,100,5] to[255,255,20].

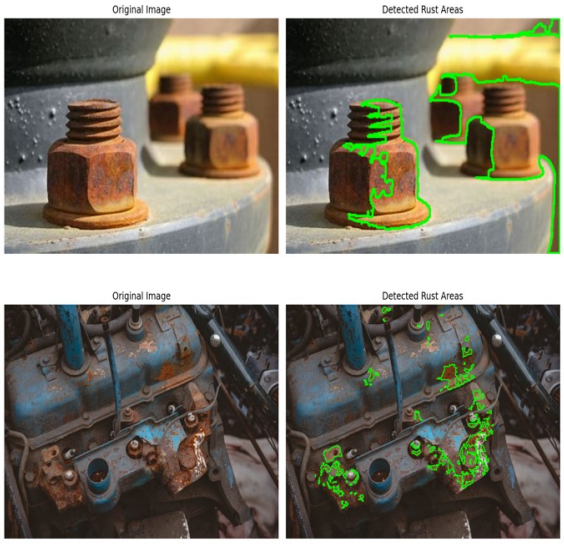


Figure 5.18: Rusty parts detection

Total images tested: 78 Correctly predicted: 73 **Accuracy**: 93.6%

- **Measurement and Dimensioning**

Similar to the approach we followed in previous hole offset calculation, I performed it step by step as we first detected the object using the square contour detection, then after the favourable contour was detected, we calculated the width and length of the square in pixels. Then the pixel distance of image width value(in pixels) and height(in pixels) is scaled to physical width and height of actual part, to calculate the actual dimensions of the part. Our pixel-to-length ratio method for measuring dimensions closely reflects the technique used by Khan et al. [7], who validated that simple dimension comparison and particle measurement can achieve reliable accuracy with minimal computational load in real-time inspection systems.

((a)) Original image ((b)) Masking background ((c)) Detecting rectangle

Figure 5.19: Measurement and dimensioning approach

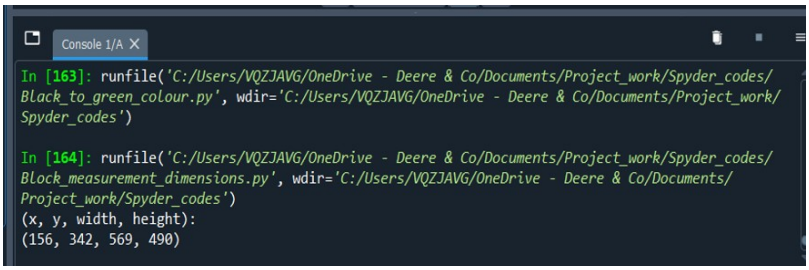
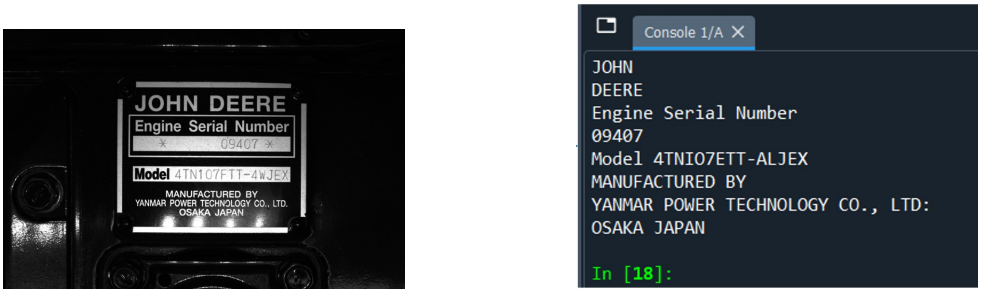

Console 1/A
In [163]: runfile('C:/Users/VQZJAVG/OneDrive - Deere & Co/Documents/Project_work/Spyder_codes/Black_to_green_colour.py', wdir='C:/Users/VQZJAVG/OneDrive - Deere & Co/Documents/Project_work/Spyder_codes')
In [164]: runfile('C:/Users/VQZJAVG/OneDrive - Deere & Co/Documents/Project_work/Spyder_codes/Block_measurement_dimensions.py', wdir='C:/Users/VQZJAVG/OneDrive - Deere & Co/Documents/Project_work/Spyder_codes')
(x, y, width, height):
(156, 342, 569, 490)

Figure 5.20: Measurement and dimensioning

- Actual Width :50 mm
- Actual Height :40 mm
- Detected Width :50.1mm
- Detected Height :39.7mm
- Accuracy:99.8%(98.5%accepted)
- Accuracy:99.25%(98.5%accepted)

- **Nameplate and barcode OCR**

OCR operations were performed to read alphanumeric characters and barcodes from engine nameplates and component labels. The EasyOCR library was used for detecting and reading printed text, as it supports a wide range of fonts and performs well on real-world industrial images. For barcode and QR code decoding, the pyzbar library was employed due to its lightweight and accurate recognition capabilities. Several other OCR tools were evaluated, but EasyOCR and pyzbar consistently provided the most reliable results in terms of speed, readability, and robustness under varying lighting and resolution conditions



((a)) Nameplate ((b)) Extacted text

Figure 5.21: OCR using EasyOCR

Figure 5.22: Detected Barcode using pyzbar

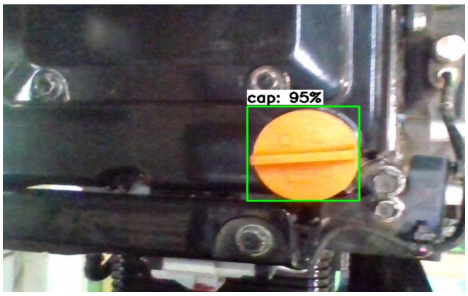
```
In [15]: runfile('C:/Users/VQZJAVG/OneDrive - Deere & Co/Documents/Project_work/Spyder_codes/
untitled18.py', wdir='C:/Users/VQZJAVG/OneDrive - Deere & Co/Documents/Project_work/
Spyder_codes')
Type: CODE128
Data: 170323YG88254
Drawing rectangle at: x=89, y=105, w=414, h=28
Type: CODE128
Data: AUC10162
Drawing rectangle at: x=90, y=48, w=285, h=34
```

Figure 5.23: Barcode decoded data

5.4 Deep Learning approach

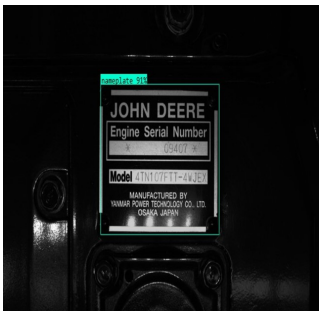
To overcome the limitations of traditional image processing techniques— as sensitivity to lighting variations, limited adaptability to complex shapes, that mostly affected object classification, presence/absence, the DL(Deep Learning)-based object detection models were explored. These models learn feature representations automatically from data, enabling more robust inspection. In this work, two architectures are implemented: SSD-MobileNetv2-FPNLite using TensorFlow(TF) and YOLOv11. Both models were evaluated for object presence /absence detection using a custom-labeled dataset captured through the R-Pi vision setup.

5.4.1 Dataset Preparation



((a)) Cap present

((b)) Cap Absent



((c)) Nameplate detection

((d)) Cap present

Figure 5.24: Dataset images for model training

Above are some of the images I used to train and test the models to compare their performance. Total Images- 168, Cap- 99, No cap- 14, Nameplate- 55.

5.4.2 SSD-MobileNet-v2-FPNLite Implementation and Results

Single Shot MultiBox Detector (SSD) is a deep learning-based object detection algorithm that performs object classification and localization in a single forward pass, making it suitable for real-time applications.The architecture of SSD-MobileNetV2-FPNLite 640×640 is optimized for real-time object detection on edge devices[14].The model uses a lightweight backbone with multi-scale feature extraction, as discussed in community forums[14]. Unlike two-stage detectors (like Faster R-CNN), SSD is faster and computationally lighter due to its single-shot architecture[15].

To balance speed and accuracy, MobileNet V2 is used as the feature extractor (backbone). It is a lightweight convolutional neural network designed for mobile and embedded vision applications. It uses depthwise separable convolutions and inverted residuals with linear bottlenecks, drastically reducing the number of parameters and computations.[16].

In summary, SSD MobileNet V2 with FPNLite provides an efficient trade-off between detection accuracy, inference speed, and resource usage, making it an ideal choice for embedded systems like Raspberry Pi or Jetson Nano where real-time performance and low power consumption are critical.

We tried testing for 3 different models as SSD-mobileNet-v2, SSD-mobileNet-v2-fpnlite and EfficientDet-Lite-D0. Among which the fpnlite model outperformed the other two.

TensorFlow Model Comparison:

Model	Accuracy(mAP)
SSD-MobileNet-v2-FPNLite	81.6%
SSD-MobileNet-v2	74%
EfficientDet-Lite-D0	59.9%

Table 5.1: Models performance comparison

Base Architecture: MobileNetV2 (as feature extractor)

- Uses inverted residual blocks
- Includes depthwise separable convolutions
- Lightweight and optimized for mobile or embedded deployment

Detection Head: SSD with FPNLite (Feature Pyramid Network Lite)

- Multi-scale feature extraction
- Enhances detection of small objects by combining low- and high-level features

Training Configuration:

- Epochs: 145
- Learning Rate: 0.004(exponential decay)
- Batch Size: 16
- Training Time: Approx 3-4 hours

mAP stands for mean average precision value

Average precision is the area under the precision-recall curve at specific IoU threshold. The mean of all Avg. precision values at different IoU(Intersection-over-Union) thresholds is called mAP.

Precision:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Recall:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

mAP at 0.70 IoU threshold- 100%

mAP at 0.75 IoU threshold- 99%

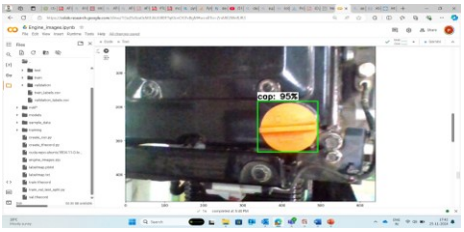
mAP at 0.80 IoU threshold- 95%

mAP at 0.85 IoU threshold- 88.89%

mAP at 0.90 IoU threshold- 27%

mAP at 0.95 IoU threshold-7%

Overall Average mAP - 81.6% for [0.50:0.95] IoU



((a)) Cap

((b)) Nameplate

((c)) No cap

Figure 5.25: SSD-MobileNet object detection

Predicted / Actual	Nameplate	No cap	Cap	False Negative
Nameplate	87.27	0	0	7
No cap	0	71.43	0	7.14
Cap	0	0	78.8	9.09
False Positive	0	21.4	12.12	0

Figure 5.26: Confusion Matrix(values in %)

5.4.3 YOLOv11 Implementation and Results

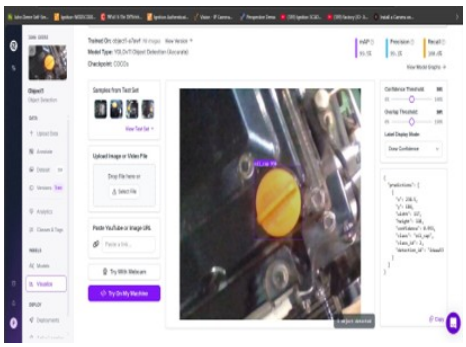
YOLO-based object detection models have shown strong promise for real-time industrial inspection tasks. Shaloo et al.[9, 17] demonstrated that a YOLOv8-based vision inspection system achieved an overall accuracy of 98 percent in classifying correctly assembled, mis-assembled, and missing parts in a flexible manufacturing line. Integrates improvements such as cross-stage partial connections, repulsion loss, and dense upsampling.

Training Configuration:

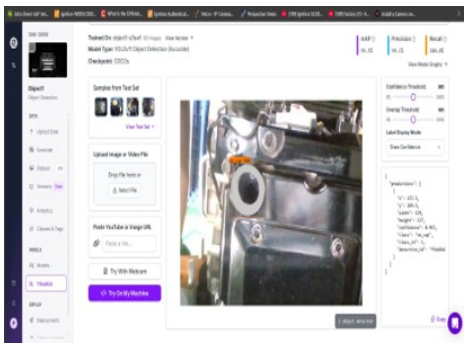
- Epochs: 150
- Learning Rate: 0.01 (Cosine decay)
- Batch Size: 16
- Training : Approx 55-80 min

Here, we used the same dataset we used in training of the SSD model so that we can transparently compare both the models when trained on the same custom dataset.

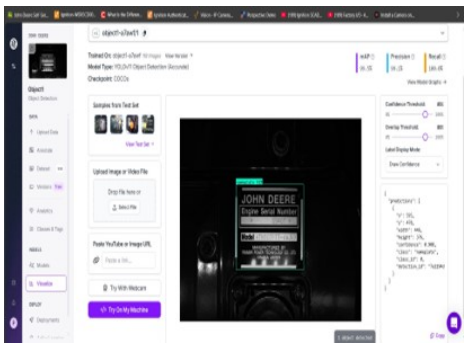
Total Images- 168, Cap- 99, No cap- 14, Nameplate- 55.



((a)) Cap present



((b)) Cap Absent



((c)) Nameplate detection

Figure 5.27: YOLOv11 object detection

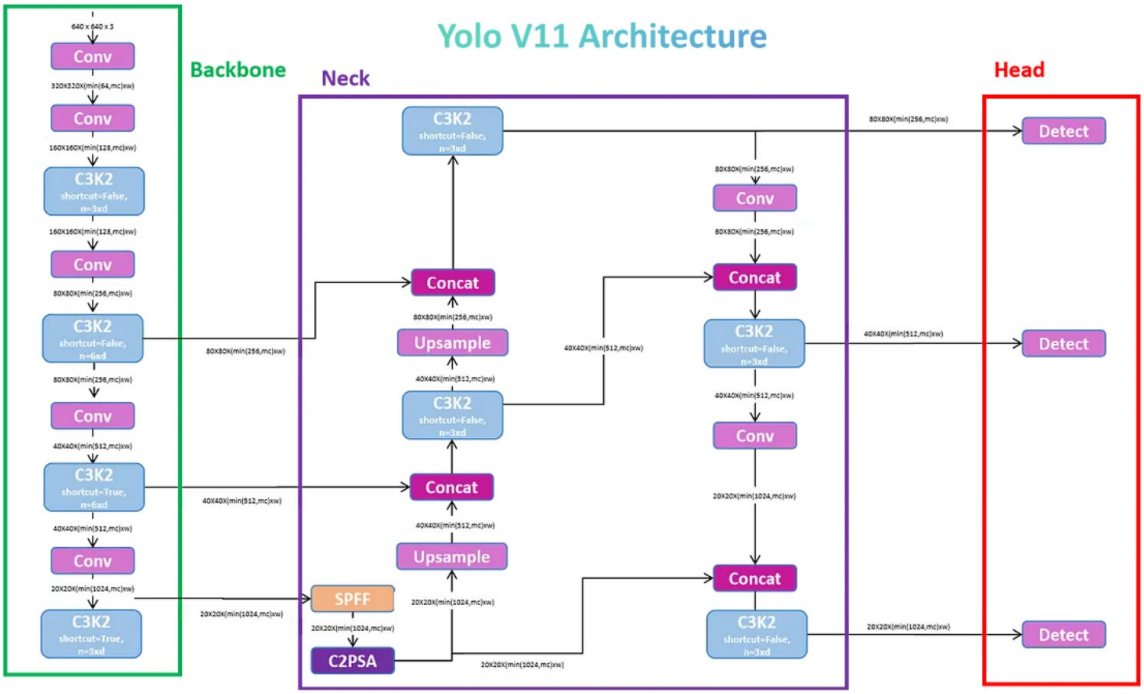


Figure 5.28: YOLOv11 Architecture [4]

Model Architecture: Composed of a Backbone, Neck, and Head

- **Backbone**: The backbone is a crucial component of the YOLO architecture, responsible for extracting features from the input image at multiple scales. This process involves stacking convolutional layers and specialized blocks to generate feature maps at various resolutions [17, 4, 18].

- **Neck**: The neck combines features at different scales and transmits them to the head for prediction. This process typically involves upsampling and concatenation of feature maps from different levels, enabling the model to capture multi-scale information effectively [9, 4, 18].

- **Head**: The head of YOLOv11 is responsible for generating the final predictions in terms of object detection and classification [17]. It processes the feature maps passed from the neck, ultimately outputting bounding boxes and class labels for objects within the image. Decodes bounding boxes and class probabilities at multiple scales[9].

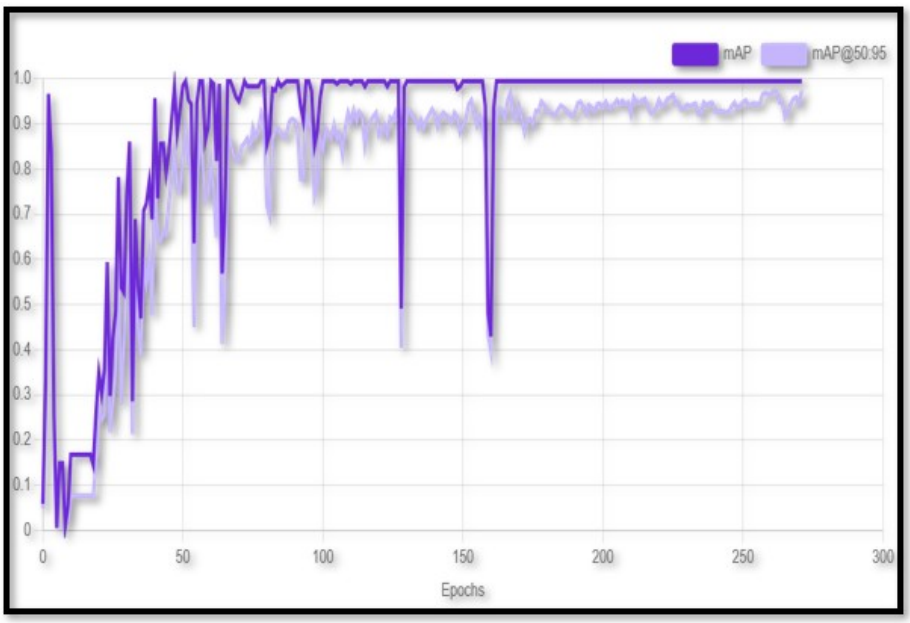


Figure 5.29: mAP

Predicted / Actual	Nameplate	No cap	Cap	False Negative
Nameplate	100	0	0	0
No cap	0	93	0	0
Cap	0	0	100	0
False Positive	0	7	0	0

Figure 5.30: Confusion Matrix(values in %)

Performance:

- Precision: 99.1%
- Recall: 99.6%
- **mAP**: 99.5%

5.5 Summary

This chapter compared three fault detection methodologies—Keyence-based inspection, image processing algorithms using Raspberry Pi kit, and deep learning models—highlighting their individual strengths and limitations:

- **Keyence Camera System**: Delivered highly reliable inspection results using inbuilt tools but involved significant cost and proprietary constraints.
- **Raspberry Pi with OpenCV**: Enabled low-cost defect detection with decent performance using methods like histogram equalization, adaptive thresholding, and edge detection; however, it struggled with object presence/absence classification.
- **Deep Learning Models**:
  - SSD-MobileNetV2-FPNLite achieved an overall mAP of 81.6%, making it suitable for embedded applications.
  - YOLOv11 outperformed other methods with a mAP of 99.5%, precision of 99.1%, and recall of 99.6%, ensuring fast and accurate real-time detection.

The findings demonstrate how deep learning significantly enhances object detection accuracy while maintaining cost-efficiency when integrated with embedded platforms like Raspberry Pi.

Chapter 6

Results and Evaluation

6.1 Introduction

This chapter presents the quantitative and qualitative evaluation of the three inspection approaches explored in this project: Keyence vision system, Raspberry Pi-based image processing and computer vision(CV), and deep learning-based object detection models. Each method is assessed based on its accuracy across multiple defect types, performance under real-world industrial conditions, and overall cost-effectiveness. The evaluation also includes a comparative analysis that identifies the most suitable inspection technique for each specific fault category. The aim is to determine the optimal approach that balances reliability, speed, and affordability for deployment in end-of-line engine inspection workflows.

6.2 Vision System Accuracy(Keyence)

This table lists the inspection accuracy achieved using the Keyence vision sensor and VS Creator software tools for different defect categories.

Fault Type	**Accuracy(in percentage)**
Presence/absence	94.78
Part Damaged	93.47
Hole Offset	93.92
Measurement dimensioning	93.04
Loose Installation	93.04
OCR	95.21
Connector Harness	93.47

Table 6.1: Accuarcy of Keyence Vision System for various defect types

6.3 Raspberry Pi based Image processing and CV

Using a low-cost Raspberry Pi kit, inspection logic was replicated using traditional computer vision techniques as histogram equalisation, thresholding, contour detection. While Wen et al.[13] primarily focused on flexible assemblies like hoses, their modular approach to preprocessing, segmentation, curve fitting helped achieve consistent accuracy, which supports the multi-stage image analysis as in our R-pi based defect detection system.

The table below shows the accuracy results for fault detection using traditional image processing and computer vision techniques implemented on a Raspberry Pi.

Fault Type	**Accuracy(in percentage)**
Presence/absence	70.2
Rust Detection	93.6
Hole Offset	99.1
Connector Harness	70.2
Measurement dimensioning	99.25
OCR	94

Table 6.2: Inspection Accuracy Using Raspberry Pi and Computer Vision Algorithms

While the Raspberry Pi approach proved effective for several defects, performance was highly dependent on lighting conditions, image clarity, and tuning parameters.

6.4 DL model Performance(TensorFlow vs YOLO11)

Deep learning models were applied specifically for part presence /absence detection using data sets captured using the Raspberry Pi kit. The work of Frustaci et al.[8] highlights that hardware-accelerated vision systems can reduce frame processing times by over 20×, demonstrating viability of deploying low-latency inspection systems without slowing down production lines—a goal similarly achieved in this project through lightweight DL models and optimized image processing logic. Two models trained and evaluated are:

- SSD MobileNet v2 fpn lite
- YOLOv11

This table compares object detection performance between the SSD MobileNet and YOLOv11 models, based on mAP, precision, and recall.

Model	mAP(%)	Precision(%)	Recall(%)
SSD mobileNet-v2-fpnlite	81	86	89
YOLOv11	99.5	99.1	99.6

Table 6.3: Performance Metrics of Deep Learning Models for object detection and classification

6.5 Comparative Analysis

Summary comparison of accuracy, cost, complexity of implementation and flexibility for Keyence, Raspberry Pi with CV, and Deep Learning-based inspection systems.

Metric	Keyence	Raspi kit+CV	DL(YOLOv11)
Cost efficiency	low	high	high
Flexibility	medium	high	high
Real-Time Capability	yes	moderate	yes
Integration ease	high	medium	medium

Table 6.4: Comparative Analysis of Inspection Approaches

Defect	Suitable model
Presence/absence	DL(YOLO)
Part Damaged	Keyence
Rust Detection	Raspi kit + CV
Loose Installation	Keyence
Connector harness	Keyence
Hole Offset	Raspi kit + CV
Measurement dimensioning	Raspi kit + CV
OCR	Keyence

Table 6.5: Defect-wise suitable inspection method

6.6 Summary

This chapter demonstrated the effectiveness and trade-offs between industrial, embedded, and AI-powered inspection solutions.

Key findings include:

- Keyence Vision System offered consistent high accuracy (more than 93%) across all defect categories, especially for surface-level faults and barcode/OCR inspection, with seamless integration but higher cost.
- Raspberry Pi with image processing and CV techniques provided a cost-effective solution, achieving high accuracy in hole offset (99.1%) and dimensioning tasks, but struggled in object classification and presence detection under inconsistent lighting.
- YOLOv11 emerged as the most accurate deep learning model for object presence/absence with a mAP of 99.5%, outperforming SSD MobileNet in both precision and recall.

The comparative analysis highlighted that a hybrid inspection system, selecting tools based on fault type, yields the most efficient and scalable solution for real-world industrial deployment.

Chapter 7

Challenges and Solutions

7.1 Introdution

This chapter discusses the key technical and practical challenges encountered during the design, implementation, and deployment of the vision-based inspection system. These challenges spanned across hardware communication, lighting sensitivity, camera alignment, software tool limitations, and the computational constraints of embedded platforms. Each obstacle is examined in the context of its impact on system performance, followed by the practical strategies and optimizations implemented to overcome them. The insights gained during this phase contributed significantly to enhancing system reliability, efficiency, and adaptability for real-world industrial use.

The development and deployment of the vision-based inspection system presented multiple challenges across hardware integration, software implementation, and model training. This chapter outlines the key obstacles encountered during the project and the practical solutions adopted to ensure a reliable and scalable inspection system as follows:

- Hardware communication and Integration Challenges
- Lighting variation and Image quality issues
- Camera positioning and field of view
- Tools and paramter sensitivity in keyence software
- Limited compute power for high-resolution, real-time inference
- Thermal management of embedded controllers during continuous operation

7.2 Hardware Communication and integration challenges

One of the critical challenges faced was achieving seamless communication between different hardware components such as the Siemens S7-1200 PLC, UR10e Co-bot, Keyence/IFM sensors, and Raspberry Pi-based systems. Various communication protocols- PROFINET for industrial components and MQTT/OPC UA for Raspberry Pi-led to some initial integration issues.

Solution:

Standardized the communication by using protocol converters and properly configuring IP addressing. PROFINET was used for deterministic communication between PLC and industrial devices, while MQTT was used for Raspberry Pi to communicate asynchronously with external systems. Careful network topology planning and diagnostic tools helped in achieving stable integration.

7.3 Lighting variation and image quality issues

Vision inspection is highly sensitive to lighting conditions. Uneven illumination, reflections, and ambient light variations affected defect detection accuracy, particularly for rust, scratch, and dent inspections. Lighting variability, limited training data, and ROI misalignment were listed by Reyna et al.[5] as ongoing challenges in MV adoption, all of which were addressed during our experimentation phase.

Solution:

Implemented diffused LED lighting and proper shielding around the inspection zones. Fixed camera mounts were used to maintain consistent angles and distances. Additionally, preprocessing techniques such as histogram equalization and adaptive thresholding were applied to normalize lighting variations.

7.4 Camera Positioning and Field of view

Incorrect or sub-optimal camera angles caused missed detections or incomplete object views, especially when inspecting irregular surfaces or cylindrical components.

Solution:

Performed a detailed analysis of part geometry and adjusted the camera mounting angle and focal length to optimize coverage. Multi-angle setups were experimented with and the best setup was locked for consistent results.

7.5 Tools and Parameter sensitivity in Keyence software

Tools such as Pattern Matching and Blob Detection in Keyence VS Creator were highly sensitive to minor part variations or manufacturing tolerances, resulting in false positives or negatives.

Solution:

Careful tuning of tool parameters like threshold, sensitivity, and Region of Interest (ROI) boundaries helped in minimizing errors. The use of ShapeTrax and Profile Defect tools allowed for more robust inspection logic in complex geometries.

7.6 Limited compute power for high-resolution, real-time inference

Running high-resolution image analysis and deep learning inference on embedded platforms like Raspberry Pi or Jetson Nano can lead to processing delays. These devices have limited GPU/CPU capabilities, which affects frame rate, especially when executing models like SSD-MobileNet or YOLO in real time.

Solution:

To address this, lightweight models such as SSD-MobileNetV2 and YOLOv11-nano were used. Inference was optimized by reducing image resolution, using quantized models (e.g., TFLite), and employing batch inference where applicable. Additionally, non-critical operations were offloaded to asynchronous threads to maintain responsiveness.

7.7 Thermal management of embedded controllers during continuous operation

Extended use of embedded systems for image processing tasks leads to increased CPU/GPU temperatures, which may throttle performance or cause system instability. This issue is critical when the device operates continuously on the shop floor.

Solution:

Thermal issues can be mitigated by installing active cooling solutions such as heat sinks and micro-fans on the processor. Additionally, periodic inference scheduling and CPU throttling mechanisms can be implemented to avoid overheating during peak loads.

7.8 Summary

This chapter addressed the major roadblocks faced in building a real-time, automated inspection solution. The key challenges and corresponding solutions included:

- **Hardware Communication**: Resolved using standardized protocols (PROFINET, MQTT) and structured network configuration.
- **Lighting Variations**: Managed with diffused lighting, shielding, and adaptive preprocessing techniques like histogram equalization.
- **Camera Positioning**: Optimized using fixed mounts and focal calibration to ensure complete coverage of inspection zones.

- **Tool Sensitivity (Keyence)**: Overcome by fine-tuning thresholds, ROIs, and adopting robust tools like ShapeTrax and Profile Defect.
- **Computational Limitations**: Mitigated through lightweight DL models, quantization, and inference optimization.
- **Thermal Management**: Handled via active cooling systems and scheduled processing logic.

These solutions ensured the smooth functioning of the system in a production environment while maintaining high accuracy and real-time responsiveness.

Chapter 8

Conclusion and Future Scope

8.1 Summary

Currently, end-of-line (EOL) engine assembly fault detection is largely dependent on manual inspection, which is time-consuming, inconsistent, and prone to human error. To address these challenges, this project successfully developed and evaluated a vision-based automated inspection system tailored for EOL fault detection. According to Javaid [6], the integration of machine vision with industrial automation not only boosts efficiency and reduces waste but also provides predictive insights, making it a foundational pillar in smart factory ecosystems. By integrating Co-bots, industrial-grade sensors like Keyence and IFM with traditional PLC-based automation, the system effectively detected defects such as presence/absence, scratches, rust, loose parts, hole offsets, and incorrect orientation.

To reduce cost and explore scalable alternatives, the project implemented a Raspberry Pi-based vision solution. The surveyed literature [5] concludes that MV systems—especially when paired with ML—are key enablers for the transition toward autonomous factories, echoing the long-term vision of this project. Using image processing and computer vision techniques, this setup was capable of detecting similar faults with acceptable accuracy, proving its viability for low-cost industrial applications. Further improvements were realized by introducing deep learning approaches using SSD MobileNet and YOLOv11, specifically for object presence and orientation detection. These models enhanced the robustness and flexibility of the inspection system under varying lighting and alignment conditions. The YOLOv8-based low-cost approach reported by Shaloo et al.[9] reinforces the potential of CNN-based models for scalable and high-accuracy real-time inspection,

aligning with the outcomes observed in our project using YOLOv11.

A comparative analysis between Keyence-based inspection, traditional image processing (OpenCV), and deep learning models demonstrated the strengths and limitations of each approach. The Keyence solution offered high accuracy and real-time performance, while the Raspberry Pi-based DL methods offered scalability and cost-efficiency. Each method contributed uniquely to building a comprehensive inspection framework.

8.2 Benefits over Manual Inspection

- Significant reduction in inspection time and manual labor.
- Improved consistency and objectivity in defect identification.
- Ability to log and trace faults using automated data collection and image storage.
- Scalability and adaptability to different part geometries and defect types.

8.3 Future Scope

While the current system provides a robust base for automated inspection, several areas offer scope for future development:

- **Real-Time Dashboard**: Integration of a live monitoring dashboard for visualizing inspection results, trends, and alerts in real time using platforms like Node-RED or Grafana.
- **Cloud Connectivity**: Enable centralized data storage and analytics through cloud services (e.g., AWS, Azure), allowing historical tracking and remote inspection monitoring.

- **Advanced Deep Learning Models**: Train and deploy more advanced and lightweight DL models optimized for edge devices to improve detection speed and accuracy.
- **Defect Severity Grading**: Move beyond binary detection (defective or not) to categorize defect severity using regression or segmentation models.
- **Integration with MES/ERP Systems**: Link inspection results with Manufacturing Execution Systems (MES) or Enterprise Resource Planning (ERP) software for automated quality control workflows.
- **Mobile Deployment**: Explore mobile inspection setups with battery-powered Raspberry Pi systems for on-the-go inspection in field service or remote environments.

REFERENCES

[1] "Raspberry pi 4 model b," "https://www.raspberrypi.com/products/raspberry-pi-4-model-b".

[2] ElProCus, "What is profinet – architecture, types and working," https://www.elprocus.com/profinet/, 2024.

[3] S. GmbH, "What is mqtt?" https://www.symestic.com/en-us/what-is/mqtt, 2024.

[4] N. Rao, "Yolov11 explained: Next-level object detection with enhanced speed and accuracy," https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f 2024.

[5] M. Reyna, G. Delgado, S. Luna, E. Chumacero, and A. Akundi, "Product digital quality inspection using machine vision systems – a categorical review," in *17th Annual IEEE Systems of Systems Engineering Conference*, 2022.

[6] M. Javaid, A. Haleem, R. P. Singh, S. Rab, and R. Suman, "Exploring impact and features of machine vision for progressive industry 4.0 culture," in *Materials Today: Proceedings*, 2023.

[7] U. S. Khan, J. Iqbal, and M. A. Khan, "Automatic inspection system using machine vision," in *Proceedings of the 34th Applied Imagery and Pattern Recognition Workshop (AIPR), IEEE*, 2005.

[8] F. Frustaci, F. Spagnolo, S. Perri, G. Cocorullo, and P. Corsonello, "Robust and high-performance machine vision system for automatic quality inspection in assembly processes," *Sensors*, vol. 22, no. 8, p. 2839, 2022.

[9] M. Shaloo, G. Princz, R. Horbe, and S. Erol, "Flexible automation of quality inspection in parts assembly using cnn-based machine learning," *Procedia Computer Science*, vol. 232, p. 2921–2932, 2024.

[10] B. E. Sciences, "Manual visual inspection is vital," https://www.boulderes.com/resource-library/manual-visual-inspection-is-vital, 2024.

[11] V. Imaging, "Enhancing manufacturing efficiency with machine vision," https://vicoimaging.com/enhancing-manufacturing-efficiency-machine-vision/, 2024, accessed: 2025-06-03.

[12] Praxie, "Automated quality inspections in manufacturing," https://praxie.com/automated-quality-inspections-in-manufacturing/, 2024.

[13] P. WEN, L. ZHENG, J. ZHOU, and B. CLARKE, "Automated inspection of flexible assembly: System design and algorithm," in *IEEE International Conference on Mechatronic and Embedded Systems and Applications*, November 2008.

[14] S. Community, "What is the architecture of ssd mobilenet v2 fpn-lite 640x640?" https://datascience.stackexchange.com/questions/116795/what-is-the-architecture-of-ssd-mobilenet-v2-fpnlite-640x640, 2024, accessed: 2025-06-03.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, F. C. Y., , and B. A. C., "Ssd: Single shot multibox detector," in *In Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[16] M. Sandler, A. Howard, A. Zhu, M.and Zhmoginov, and Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] Y. Authors, "Yolov11: An overview of the key architectural enhancements," https://arxiv.org/html/2410.17725v1, 2024.

[18] TutorialsPoint, "Building a real-time object detection system with yolo algorithm," https://www.tutorialspoint.com/building-a-real-time-object-detection-system-with-yolo-algorithm, 2024, accessed: 2025-06-03.